

ElectricFlow 6.0.4 User Guide

Electric Cloud, Inc.
www.electric-cloud.com

ElectricFlow 6.0

Copyright © 2002 – 2015 Electric Cloud, Inc. All rights reserved.

Published 3/31/2016

Electric Cloud® believes the information in this publication is accurate as of its publication date. The information is subject to change without notice and does not represent a commitment from the vendor.

THE INFORMATION IN THIS PUBLICATION IS PROVIDED “AS IS.” ELECTRIC CLOUD, INCORPORATED MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIMS IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Use, copying, and distribution of any ELECTRIC CLOUD software described in this publication requires an applicable software license.

Copyright protection includes all forms and matters of copyrightable material and information now allowed by statutory or judicial law or hereinafter granted, including without limitation, material generated from software programs displayed on the screen such as icons, screen display appearance, and so on.

The software and/or databases described in this document are furnished under a license agreement or nondisclosure agreement. The software and/or databases may be used or copied only in accordance with terms of the agreement. It is against the law to copy the software on any medium except as specifically allowed in the license or nondisclosure agreement.

Trademarks

Electric Cloud, ElectricAccelerator, ElectricCommander, ElectricInsight, and Electric Make are registered trademarks or trademarks of Electric Cloud, Incorporated.

Electric Cloud products—ElectricAccelerator, ElectricCommander, ElectricInsight, and Electric Make—are commonly referred to by their “short names”—Accelerator, Commander, Insight, and eMake—throughout various types of Electric Cloud product-specific documentation.

Other product names mentioned in this guide may be trademarks or registered trademarks of their respective owners and are hereby acknowledged.

Contents

Introduction to ElectricFlow	1
Web-Based System	1
Automation Platform	40
What Makes ElectricFlow Unique?	40
ElectricFlow Architecture	41
Simple Architectural Overview	41
Expanded Remote Configuration	42
Other Configurations	43
Build-Test Automation	43
Deployment Automation	47
Pipeline Automation	51
Roadmap to ElectricFlow	51
ElectricFlow Glossary	53
Build-Test Automation	77
Getting Started on Build-Test Automation	77
Build-Test Automation Concepts, Features, and Functionality	77
Roadmap for Build-Test Automation Tasks	80
Deployment Automation	81
Getting Started with Deployment Automation	81
Deployment Automation Concepts and Objects	82
What You Can Do for Deployment Automation	84
One Application for Multiple Use Cases	85
Keeping Users Up-To-Date	85
Keeping Track of Changes	85
Taking Snapshots	85
Optimizing How Resources are Used	86
Customizing Deployments	86
Getting the Real-Time Status of Application Runs and Troubleshooting	87
Modeling and Deploying Applications in Static Environments	87
Logging in to ElectricFlow	89

Modeling Applications	90
Creating an Application and Application Tiers	91
Creating a New Application	92
Adding Components to the Application Tiers	93
Adding a Component	95
Adding an Application Tier	95
Modeling Component Processes	95
Modeling More Component Processes	102
Modeling Application Processes	102
Modeling More Application Processes	110
Using the Drag and Drop Method to Add Process Steps	110
Setting Parameters for Application Processes	113
Setting and Modifying the Parameter Label	121
Looking Up Parameters	126
Adding Credentials	127
Using Plugins	128
Setting Email Notifications	129
Configuring recipients	133
Configuring the event that triggers the notification	134
Configuring the environments where the notification applies	134
Selecting and Editing Email Messages	137
Component and Application Process Steps: ectool Example	142
Modeling Environments	146
Adding an Environment	146
Assigning Resources to Environment Tiers	148
Making Tier Maps	150
Deploying (Running) Applications	152
Overview	153
Deploying an Application	154
Running Applications with New Parameters	155
Deploying Applications with Parameters from Previous Runs	162

Deploying (Running) Applications with Schedules	168
Deploying Snapshots	175
Deploying or Comparing Snapshots	177
Setting Parameters in the Dialog Box	177
Deploying Snapshots	178
Comparing Snapshots	178
Deploying Applications with Parameters	178
Examples: Deploying (Running) Applications	179
Full Run	179
Partial Runs with Specific Artifact Versions	180
Viewing Results and Troubleshooting	182
Viewing the Environment Inventory and Application Inventory	183
Examples: Viewing Deployment Details	185
Example 1: Application Process Level Details	186
Example 2: Application Process Step Level Details	187
Example 3: Component Process Step Details	187
Example 4: Change History of a Resource	188
Searching the Change History	188
Modeling and Deploying Applications in Dynamic Environments	191
Logging in to ElectricFlow	194
Modeling Applications	195
Creating an Application and Application Tiers	196
Creating a New Application	197
Adding Components to the Application Tiers	198
Adding a Component	200
Adding an Application Tier	200
Modeling Component Processes	200
Modeling More Component Processes	207
Modeling Application Processes	207
Modeling More Application Processes	215
Using the Drag and Drop Method to Add Process Steps	215

Setting Parameters for Application Processes	218
Setting and Modifying the Parameter Label	226
Looking Up Parameters	231
Adding Credentials	232
Using Plugins	233
Setting Email Notifications	234
Configuring recipients	238
Configuring the event that triggers the notification	239
Configuring the environments where the notification applies	239
Selecting and Editing Email Messages	242
Modeling Dynamic Environments	247
Usage Guidelines and Best Practices	248
Creating AMIs	249
AMIs with ElectricFlow Agents	249
AMIs with ElectricFlow Agents and Chef Configuration Management	250
Creating Resource Templates	251
Configuring Amazon EC2 as the Cloud Provider	252
Configuring OpenStack as a Cloud Provider	257
Configuring Chef as the Configuration Management Tool	264
Viewing and Editing Resource Templates	268
Resource Template Details	268
Resource Template Properties	271
Access Control Settings	272
Change Tracking	272
Resources Page	272
Resource Pools Page	273
Accessing the Resource Templates in the Automation Platform	274
Creating Environment Templates	276
Creating a New Environment Template	276
Creating an Environment Template Based on an Existing Template	281

Viewing and Editing Environment Templates	283
Environment Template Details	283
Modeling Environments with Resources or Resource Pools	288
Developer Task: Creating Custom Plugins	292
Creating Custom Cloud Provider Plugins	293
How to Create a Custom Cloud Provider Plugin	293
Example: Property Structure for a Cloud Provider Plugin	296
Creating Custom Configuration Management Plugins	297
How to Create a Custom Configuration Management Plugin	298
Example: Property Structure for a Configuration Management Plugin	301
Deploying Applications With Provisioned Cloud Resources	302
Deploying Applications with Parameters	310
Retiring Dynamic Environments	311
Dynamic Environment Example with Amazon and Chef	312
Resource Templates	312
Environment Templates	313
Deploying Applications to Dynamic Environments	315
Retiring Dynamic Environments	315
Developer Task: Creating Custom Plugins	316
Creating Custom Cloud Provider Plugins	317
How to Create a Custom Cloud Provider Plugin	317
Example: Property Structure for a Cloud Provider Plugin	320
Creating Custom Configuration Management Plugins	321
How to Create a Custom Configuration Management Plugin	322
Example: Property Structure for a Configuration Management Plugin	325
Master Components	326
Creating a New Master Component	327
Creating a Master Component Based on an Existing Master Component	327
Creating a Master Component Based on an Existing Application Component	328
Adding a Component Process to a Master Component	328
Editing a Master Component	329
Creating an Application Component Based on a Master Component	329
Creating an Application Component Based on an Existing Application Component	329

Deleting Master Components	330
Master Components List UI	330
Deploying Applications in ElectricFlow	332
Deployment Automation User Interface	333
ElectricFlow Buttons and Icons	333
Landing Page	348
Home Page	348
Main Menu	349
Applications	350
Environments	350
Automations	351
Admin	352
Applications List	354
Applications Visual Editor	355
Application Tiers	356
Context Menu for the Application Tier and Component	356
In the Application Tier	357
In the Component	357
Defining Components	360
Creating Component and Component Processes	363
Component Processes	363
Component Process Visual Editor UI	366
Application Processes	369
Application Process Visual Editor UI	370
Example: Defining Process Steps	372
Component and Application Process Steps	375
Designing a New Process Step	375
Modeling an Existing Process Step	376
Setting Parameters in the Process Step Dialog Boxes	376
Parameters in Application Processes	377
Environments List	380
Environments Visual Editor	383
Environment Tiers	385
Context Menu for the Environment Tier	386

Context Menu for the Resource	387
Adding Resources to an Environment Tier	388
Environment Templates List	388
Resource Templates List	389
Tier Maps	390
Application Notifications Dialog Box	392
Setting Notifications for the First Time	393
Enabling Notifications	394
Adding Notifications	396
Configuring recipients	396
Configuring the event that triggers the notification	397
Configuring the environments where the notification applies	397
Adding More Notifications	398
Deleting Notifications	398
Disabling Notifications	399
Applications	399
Application Processes and Process Steps	400
Change History Search Form	401
Change History Page	402
Paths to Objects	404
Detailed Object Changes	404
Snapshot List	405
Snapshot Details Page	407
Snapshot Dialog Boxes	409
New Snapshot Dialog Box	410
Component View in the New/Preview Dialog Box	410
Modifying the Artifact Version	413
App Process View in the New/Preview Dialog Box	413
Inventory Tracking	414
Tracking at the Component Process Level	414
Application Inventory	415
Environment Inventory	417

Viewing Job Details	420
Change Tracking	423
Performance Consequences of Change Tracking	424
Use Case 1: Importing and Cloning Large Projects with Change Tracking Enabled	425
Use Case 2: Change Tracking of Non-Project-Owned Objects	426
Estimating Database Growth	426
Common Usage Patterns	427
Estimating Process	427
Alternative Estimating Methods	429
Best Practices for Change Tracking	429
When to Enable or Disable Change Tracking for a Project	429
Splitting Tracked Objects Into Separate Projects	430
Configuring Change Tracking	431
Enabling Change Tracking Globally	431
Enabling Change Tracking on a Per-Project Basis	431
In the ElectricFlow Platform UI	431
Using ectool	432
Using ec-perl:	432
Upgrading to ElectricFlow 6.x	433
Customizing the Change History Page	433
Searching the Change History	433
Modifying the Change History	436
Change History Time Line	436
Default Settings	437
Number of Changes	437
Time Increment	438
Moving the Start and End Times Manually	439
Setting Custom Time Increments	439
Change History Filters	442
Viewing the Change History	443
Viewing the Change History from the Applications List	444

Viewing the Change History From an Application or Environment	445
Application Visual Editor	445
Component Process Visual Editor	448
Environments Visual Editor	450
Environment Tier	451
Viewing the Change History for Artifacts, Jobs, Projects, and Workflows	451
Artifacts	452
Jobs	453
Projects	454
Workflows	454
Change History Page	454
Time Line	457
Default Settings	457
Selecting the Time Range	458
Moving the Start and End Times	459
Specifying the Time Range	460
Number of Changes	463
Paths to Objects	464
Detailed Object Changes	464
Filters	464
Reverting Changes to Objects	466
Credentials and User Impersonation	468
What is a Credential?	468
Defining a Credential	469
Why Use Credentials?	469
Credential Access Control	469
Using Credentials for Impersonation	470
Setting the Impersonation Credential in the Automation Platform	471
Attaching Impersonation Credentials to Steps, Procedures, and Projects	471
Attaching Impersonation Credentials to Schedules	471
Attaching Impersonation Credentials to Jobs	471
Attaching a New Impersonation Credential in the Automation Platform	471

Accessing Credentials From a Step	472
Attaching a Credential to a Procedure Step, Procedure, or Schedule	472
Passing Credentials as Parameters	473
Credential References	473
Best Practices for Retrieving Credentials	474
First Class	474
Second Class	474
Does the step's command use properties?	474
Does the step execute code or commands that can be modified?	474
Avoid passing passwords on the command line	475
Attaching Credentials in Deployment Automation	475
Example	475
Process Branching	476
About Process Branching	477
How to Use the Process Branching UI	478
UI Objects	478
Branching Conditions Menu	480
Analysis of a Process With Simple Branching	483
Process Branching States and Conditions	485
485	
State of the Branching Condition Connectors in the UI	485
Examples of Branching Conditions	485
Custom Conditions in Process Branching	486
Configuration Guidelines for Process Branching	490
Simple Process Branching Example	490
Process Branching Example: Deleting Steps	497
Deleting Step 3	499
Deleting Step 7	501
Deleting Step 2	502
Adding Snapshots	503
Deploying Snapshots	507
Deploying or Comparing Snapshots	508

Setting Parameters in the Dialog Box	509
Deploying Snapshots	510
Comparing Snapshots	510
Managing Snapshots	510
Comparing Snapshots	511
Comparing an Application to Snapshots	512
Comparing Snapshots	514
Comparing an Application to Snapshots	516
Comparing Snapshots in the Snapshot List	519
Deleting Snapshots	522
Selecting Snapshots to Delete	522
Deleting All the Snapshots	523
Automation Platform Tasks	524
Pipeline Automation	525
Automation Platform Basics	526
What the Automation Platform Does	526
Getting Started	529
Overview	529
Basic Automation Platform Terminology	529
Getting Started Scenarios Help Topic Series	529
Navigating the Automation Platform User Interface	530
In addition to the main set of tabs, the top bar includes:	530
Home Tab	531
Continuous Integration Subtab	532
Projects, Jobs, and Workflow Tabs	533
Cloud Tab	534
Resources Page	534
Pools Subtab	534
Workspaces Subtab	535
Zones and Gateways Subtabs	535
Artifacts and Search Tabs	535
Artifact Versions Subtab	535

Repositories Subtab	536
Search Tab	536
Administration Tab	536
Getting Started - Scenario 1 - Creating a Simple Procedure	537
Overview	538
Begin Scenario 1	538
Scenario Extension - Adding Another Step	544
Finding an ElectricFlow Automation Platform Web Page	545
Summary	546
Getting Started - Scenario 2 - Creating a Procedure That Uses an SCM	546
Overview	548
Begin Scenario 2	548
Scenario extension - Add a step to show workspace contents	556
Summary	557
Getting Started - Scenario 3 - Notification, Scheduling, and Reporting	557
Overview	559
Begin Scenario 3	559
Scenario extension - Add a thumbnail report to your Home page	569
Summary	571
Getting Started - Scenario 4 - Multi-agent Build and Test	571
For Linux	572
For Windows	572
Overview	572
Begin Scenario 4	573
Scenario extension - Using postp	578
Summary	580
Automation Platform Setup	580
Automation Platform Home Page	580
Overview	581
Job Configurations	581
Create Job Configurations three ways	581

Shortcuts	582
Create Shortcuts two ways	582
Jobs Quick View	582
Create a job category	582
Reports	582
Customizing the ElectricFlow Platform UI	583
Customizing parameters	583
How do you customize parameters?	583
Custom parameter form contents	583
Example	584
Customizing the tab layout	585
Overview	585
View definition syntax	586
Storing views	588
Default views	588
Developing and troubleshooting	588
Home page configuration	589
User settings	589
Location	589
Installing the Home page	589
Reconfigure Contact Support link	589
Configuring ElectricFlow	590
Web Interface Online Help System	591
Setting Up Resources	591
What is a resource?	591
ElectricFlow supports two resource types	592
Standard Resources	592
Proxy Resources	592
Setting up SSH keys	593
To create a new resource	593

Gateways and zones	593
What is a zone?	593
Cross-zone communication	594
What is a gateway?	594
Support for gateways	595
Setting Up Workspaces	595
To create a new workspace	595
Setting Up Email Configurations	595
To create an email configuration	595
Email Notifications in ElectricFlow	596
Setting Up a Source Control Configuration	596
Setting Up Directory Providers	597
To specify a directory provider	597
Enable/Disable Local ElectricFlow Users	597
wrapper.conf Properties	598
Using ElectricFlow in Your Environment	598
What's in a step?	598
Where's the script for a step?	599
Two ways to handle script execution	599
Two cases where it makes sense to store the script for a step in the ElectricFlow step	599
Environment variables	600
When do you use subprocedures?	600
How do you evolve procedures?	601
Porting existing scripts	601
ElectricFlow project version control	602
ElectricFlow Installed Tools	602
Usage	603
Commands	603
Server Communication Options	606
Global Options	606

Examples	606
Example 1: Configure an agent to talk to any server (untrusted mode)	606
Example 2: Configure an agent to accept connections only from a single remote ElectricFlow server	606
Example 3: Configure an ElectricFlow server with additional host names in the certificate	607
Usage	607
ElectricFlow agent configuration options	607
Apache web server configuration options	608
ElectricFlow Server configuration options	609
Repository Server configuration options	610
General options	611
Examples	611
Use Cases	612
Using ecdaemon	612
Command-line parsing	612
ec-perl considerations	613
ecproxy Algorithm	613
ecproxy Operations	613
getDefaultWorkingDirectory	613
getDefaultTargetPort	614
connect	614
uploadFile	615
generateWrapperScript	615
uploadWrapperScript	615
generateWrapperInvocationCommand	616
runCommand	616
cleanup	617
ping	617
Available Helper Functions	618
mesg	618
readFile	618

writeFile	618
initDispatcher	618
setOperation	619
loadFile	619
setSSHKeyFiles	619
setSSHUser	619
useMultipleSSHSessions	620
Examples	620
"Real World" Examples	621
ClusterExec	621
MySQL	622
Android	623
Setting up the process	625
Creating a Setup Step	625
Update the Postprocessor field for steps in your procedure	625
Add a Final Step to your procedure	625
Copying Other Files from the Workspace	626
Prerequisites	627
Command	627
Import Files	627
Prerequisites	628
Locations	628
Command	628
Output	628
Automation Platform Tasks	629
Access Control	630
Reading and Using This Page	630
Access Control - defining entries	631
Privileges - create new or edit existing privileges	632
Artifacts	632

Artifact Details	633
General Information section	633
The "tabbed" sections	634
Artifact Versions table	634
Properties table	634
Artifact - create new or edit existing artifact	634
To create a new artifact	635
To edit an artifact	635
Artifact Versions	635
Artifact Version Details	636
General Information section	637
The "tabbed" sections	637
Files	637
Retrievers table	637
Dependent Artifact Versions table	638
Properties table	638
Artifact Version - edit an existing artifact version	638
To edit an artifact version	638
Repositories	639
Repository - create new or edit existing repository	640
To create a new repository	640
To edit a repository	641
Database Configuration	641
Defect Tracking Configurations	642
Link at the top of the table	642
Column descriptions	642
Defect Tracking - create new or edit existing configuration	643
To create a defect tracking configuration	643
Using the defect tracking integration	643
To edit an existing defect tracking configuration	644
Defect Tracking Reports	644

Email Notifier - create new or edit existing email notifier	644
For jobs and job steps - two types of email notifiers:	644
For workflow states - three types of email notifiers:	645
Using email notifiers	645
To create a new email notifier	645
To edit an existing email notifier	647
Email notifier templates	647
Email Configurations	649
Email Configuration - create new or edit existing email configuration	649
To create a new email configuration	649
To edit an existing email configuration	650
Email Notifier - create new or edit existing email notifier	650
To create a new email notifier	651
To edit an existing email notifier	653
Email notifier templates	653
Email Notifier Template - Html_JobStepTempl_AllSteps.txt	655
Email Notifier Template - Html_JobStepTempl_SingleStep.txt	656
Email Notifier Template - Html_JobTempl.txt	657
Email Notifier Template - Html_StateTemplate_ApproveWorkflow.txt	658
Email Notifier Template - Html_StateTemplate_FullWorkflow.txt	659
Email Notifier Template - JobStepTempl_FullProps.txt	660
Email Notifier Template - JobTempl_FullProps.txt	661
Email Notifier Template - StateTemplate_FullPropertyPaths.txt	662
Event Log	663
Column descriptions	664
Configuring the Event Log	664
Automation Platform Home Page	664
Overview	665
Job Configurations	665
Create Job Configurations three ways	665

Shortcuts	665
Create Shortcuts two ways	666
Jobs Quick View	666
Create a job category	666
Reports	666
Automation Platform Home Page	667
Overview	667
Job Configurations	667
Create Job Configurations three ways	668
Shortcuts	668
Create Shortcuts two ways	668
Jobs Quick View	668
Create a job category	668
Reports	669
Job Configuration	669
To create a new job configuration	669
To edit an existing job configuration	670
Shortcuts	670
Jobs Quick View	671
To create a new jobs quick view category	671
To edit an existing jobs quick view category	672
New Report	672
To populate the drop-down menu	672
Jobs	673
Links at the top of the table	673
Column descriptions	673
Tips	674
Job Step Details	674
Links and actions at the top of the table	674
Summary section (at the top of the page)	674

The "tabbed" Sections	676
Child Steps table	676
General table	676
Diagnostics table	677
Parameters table	677
Properties table	677
Notifiers table	677
Job Step Time and waitTime Properties Explained	678
Job Details	679
Links and actions at the top of the table	679
Summary section (at the top of the page)	680
The "tabbed" sections	681
Steps table	681
Diagnostics table	682
Parameters table	683
Properties table	683
Notifiers table	684
Published Artifact Versions table	684
Column descriptions	685
Retrieved Artifact Versions table	685
Column descriptions	685
Licenses	685
All Licenses	685
Column descriptions	686
Current Usage	686
License Types	687
Concurrent Resource License	687
Concurrent User License	688
Concurrent Step License	688
Registered Host License	688

View License	689
Import License	690
Plugin Manager	690
Tab and field descriptions above the table	690
Column descriptions	691
Using a plugin	692
Configuring the plugins directory	693
Procedure Details	693
Procedure Steps	693
Creating a new step	694
Parameters	694
Email Notifiers	695
Custom Procedure Properties	695
Procedure - create new or edit existing procedure	695
To create a new procedure	695
To edit an existing procedure	697
Step - create new or edit existing step	697
To create a new command or subprocedure step	697
To edit an existing command or subprocedure step	703
Publish Artifact Version Step	704
Retrieve Artifact Version Step	704
Artifact Retrieval Dependency Order Explained	706
Send Email Step	707
New Extract Preflight Sources Step	708
Parameter - create new or edit existing parameter	708
To create a new parameter	709
To edit an existing parameter	711
Projects	711
Projects have two purposes	712
ElectricFlow includes default projects	712

ElectricFlow Solutions	713
Project - Create New or Edit Existing Project	713
Creating a New Project	713
Editing an Existing Project	714
Project Details	715
The "tabbed" sections	716
Procedures tab	716
Workflow Definitions tab	716
Jobs tab	717
Workflows tab	717
Schedules tab	718
Credentials tab	719
Properties tab	719
Reports tab	719
Run Procedure	720
Using the Run Procedure page	720
Schedule - create new or edit existing schedule	721
To create a new standard schedule	721
To create a new continuous integration "schedule"	724
To edit an existing schedule	724
Credentials - create new or modify existing credential	725
To create a new credential	725
To edit a credential	725
Properties	726
Overview	726
Creating or modifying properties	727
Using property values	728
Property sheets and intrinsic properties	728
Property names and paths	729
Absolute property paths	729

Relative property paths	731
Context-relative "shortcuts" to property paths	732
Property path shortcuts in ElectricFlow 5.0 and later	736
Property name substitutions	737
Expandable properties	738
Custom property names and values	738
The property hierarchy	739
Special property references	740
increment	740
timestamp	740
javascript	741
ElectricFlow Intrinsic Properties listed by object type	742
ElectricFlow Property Type definitions	743
ElectricFlow Object / Property tables	744
Property error codes	806
Property - create new or edit existing property	808
Nested Property Sheet	809
Reports	809
Create a new report	810
Multiple Series Report	810
Category Report	811
Count Over Time Report	812
Procedure Usage Report	813
Resources	814
Supported Resource Types	815
Resource Page Information and Functions	816
Links, Icons, and Buttons Above the Table	816
Table view - column descriptions	818
Grid view	821
Filters pane	822

New Resource panel	823
New Proxy Resource panel	827
Edit Resource panel	831
Edit Proxy Resource panel	831
Resource Details panel	831
Switching a non-trusted agent to trusted	832
Manually switching a non-trusted agent to trusted status	832
Install or Upgrade Remote Agents	832
Prerequisites:	833
Using the Install or Upgrade Remote Agents dialog	833
Parameters	835
Advanced	836
Resource Pools	840
Resource Pool - create new or edit existing pool	841
To create a new resource pool	841
To edit an existing resource pool	843
Unusable resources	843
Resource Pool Details	844
The "tabbed" sections	844
Properties tab	846
Zones	846
Zone Details panel	847
Creating a new zone	847
Access Control notes	848
Gateways	848
Gateway Details panel	850
Create Gateway panel	850
Edit Gateway panel	851
Access Control note	851
Accessing the Resource Templates in the Automation Platform	851

Define Search	853
Search Results	854
Server	855
Settings - edit existing property settings	855
Source Control Configurations	856
Source Control Configurations - create new or edit existing configuration	856
AccuRev	857
ClearCase	857
File	858
Git	858
Perforce	858
Property	859
Subversion	859
Checking out code from a source control system	859
To edit an existing source control configuration	860
Active Users	860
Users and Groups	860
User - create new or edit existing local user	862
To create a new local user	862
To edit an existing local user	862
User Details	863
Edit User Settings	863
Groups	864
Group - create new or edit existing local group	864
To create a new local group	864
To edit a local group	864
Group Details	864
Directory Providers	865
Directory Providers - create new or edit existing directory providers	866
ElectricFlow supports Active Directory and LDAP directory providers	866

To create a new Active Directory provider	866
To create a new LDAP directory provider	869
Examples for directory provider field descriptions	872
To edit an existing directory provider	874
Test Directory Provider	874
Workflows	875
Workflow Definition - create new or edit existing workflow definition	875
To create a new workflow definition	875
To edit an existing workflow definition	876
Workflow Definition Details	876
Graph view	878
To create a new state	879
To configure the new State Definition	880
To create a Transition Definition	884
Using the graph's "quick-access" features	886
Show Legend	887
List view	888
Properties view	890
Run Workflow	891
Workflow Details	892
Summary section - at the top of a running workflow page	893
When the workflow is complete... ..	894
Graph tab	895
Show Legend tab	895
Using the graph's "quick-access" features	896
Show History	896
States tab	898
State Details panel	899
Parameters tab	902
Properties tab	902

Workflow Log	903
Transition Workflow	904
Workspaces	905
Workspace - create new or modify existing workspace	906
To create a new workspace	906
To edit an existing workspace	908
Workspace File	908
ElectricFlow Platform Objects and Functionality	908
Access Control	909
Overview	909
Privileges	909
Users and Groups	910
Special Users and Groups	910
Access Control Lists - allow and deny	910
Inheritance	911
Additional Information about "deny"	911
System Objects	912
Access Control and Jobs	913
Examples for Increased Security	914
Abbreviations used in the examples	914
Example 1 - Basic ACL Setup	914
Server ACLs	915
Custom Server Properties ACLs	915
System Object ACLs	915
Plugin Project ACLs	917
Project ACLs	917
Example 2 - Team ACL Setup	918
Server ACLs	919
Custom Server Properties ACLs	919
System Object ACLs	919
Plugin Project ACLs	921

ACLs for Existing Projects	921
ACLs for New Projects	922
Optional: Restricting Resources and Workspaces by Team	924
Resource ACLs	924
Workspace ACLs	925
Artifact Management	926
Overview	926
Artifact Objects	926
Artifact	927
Examples	927
Artifact Version	927
Version Strings	928
The artifactVersionState property	929
Repository	930
Using multiple repositories	932
Configuring the Artifact Repository Server for Amazon Simple Storage Service	932
The repositoryDisabled intrinsic property	934
Access Control	934
Publishing Artifact Versions	936
Enable Compression	937
Include and Exclude Patterns	937
Retrieving Artifact Versions	937
Entering Filters During a Retrieve Operation	939
Dependent Artifact Versions	941
Artifact Cache	942
Cleaning Up Repositories and Caches	943
Authenticating Users for LDAP and Active Directory	944
Configuring LDAP	944
Sample LDAP Configuration	944
The following properties configure LDAP mapping:	945

Determining LDAP Mapping	946
Sample LDAP User Record	947
Sample LDAP Group Record	947
Sample Active Directory Configuration File	947
Change Tracking	949
Configuring Change Tracking	950
Enabling Change Tracking Globally	950
Enabling Change Tracking on a Per-Project Basis	950
In the ElectricFlow Platform UI	951
Using ectool	951
Using ec-perl:	951
Upgrading to ElectricFlow 6.x	952
Customizing the Change History Page	952
Viewing the Change History for Artifacts, Jobs, Projects, and Workflows	953
Artifacts	953
Jobs	954
Projects	955
Workflows	956
Modifying the Change History	956
Change History Time Line	956
Default Settings	957
Number of Changes	957
Time Increment	958
Moving the Start and End Times Manually	959
Setting Custom Time Increments	959
Change History Filters	962
Searching the Change History	963
Reverting Changes to Objects	966
Change History Search Form	968
Change History Page	969

Paths to Objects	971
Detailed Object Changes	972
Credentials and User Impersonation	972
What is a Credential?	973
Defining a Credential	973
Why Use Credentials?	973
Credential Access Control	974
Using Credentials for Impersonation	974
Setting the Impersonation Credential in the Automation Platform	975
Attaching Impersonation Credentials to Steps, Procedures, and Projects	975
Attaching Impersonation Credentials to Schedules	975
Attaching Impersonation Credentials to Jobs	975
Attaching a New Impersonation Credential in the Automation Platform	976
Accessing Credentials From a Step	976
Attaching a Credential to a Procedure Step, Procedure, or Schedule	977
Passing Credentials as Parameters	977
Credential References	977
Best Practices for Retrieving Credentials	978
First Class	978
Second Class	978
Does the step's command use properties?	978
Does the step execute code or commands that can be modified?	979
Avoid passing passwords on the command line	979
Defect Tracking	979
Scenario example	980
Example: Enabling the JIRA integration in your procedure	980
ElectricSentry	981
How ElectricSentry works	982
Configuring ElectricSentry	982
Quiet time	982

Resource	983
Polling frequency	983
Time-of-day and day-of-week	983
Configuring a build for continuous integration	983
Source Control Management (SCM) configurations	983
Create a procedure	984
Create a schedule	984
Optional - running ElectricSentry on multiple resources	984
Overview	984
Configuring ElectricSentry to use multiple resources	985
The Job Step Execution Environment	985
Terms and definitions	986
Preflight Builds	988
Why use Preflight Builds?	988
Preflight Build Solution	988
Preflights with ElectricFlow	989
Workflow	989
Components	989
Installation	989
Configuration	990
Server	990
Agent	991
Client	991
Samples	991
Default SCMs	996
Perforce	997
Subversion	998
AccuRev	998
ClearCase	998
Bazaar	999

Git	999
CVS	1000
Mercurial	1000
Repo	1000
StarTeam	1001
TFS	1001
Vault	1001
Troubleshooting	1002
Client	1002
Agent	1002
Properties	1002
Overview	1002
Creating or modifying properties	1004
Using property values	1004
Property sheets and intrinsic properties	1005
Property names and paths	1005
Absolute property paths	1006
Relative property paths	1008
Context-relative "shortcuts" to property paths	1009
Property path shortcuts in ElectricFlow 5.0 and later	1012
Property name substitutions	1014
Expandable properties	1014
Custom property names and values	1015
The property hierarchy	1015
Special property references	1016
increment	1016
timestamp	1017
javascript	1017
ElectricFlow Intrinsic Properties listed by object type	1019
ElectricFlow Property Type definitions	1020

ElectricFlow Object / Property tables	1020
Property error codes	1082
Postprocessors: Collecting Data for Reports	1084
Overview	1084
Postp	1085
Extending postp: matchers	1085
Postp functions	1087
Integration with the ElectricFlow user interface	1088
Custom property names and values	1089
Diagnostic information	1089
Postp integration with Java Tools	1090
The process... ..	1090
From this output... ..	1091
Java Tool matcher examples	1091
Two examples of postp Emma matchers:	1091
An example of postp JUnit matchers:	1092
An example of postp Clover matchers:	1092
Artifacts directory	1092
Reports	1093
Run reports on a non-local resource	1093
Real-time reports	1093
Home page	1093
Jobs	1094
Job details / Step details	1094
Resources	1094
Build reports using ecreport	1094
Example 1	1096
Example 2	1097
Default Batch Reports (Run Reports)	1097
Report types	1097

Default Batch Reports	1098
Tip	1099
Advanced reporting information - ecrptdata	1100
How are default batch reports generated?	1100
Filtering	1101
Adding additional data columns to the report	1102
Secure login	1102
Optional Batch Reports	1103
Creating a Report Job	1103
Viewing the report	1105
Optional Batch report examples	1105
Category sample report	1105
Procedure Usage Sample Report	1107
Count Over Time Sample Report	1108
Multiple Series Report	1111
Creating Custom reports	1114
Data extraction	1115
Using ectract.pl for data extraction	1115
Examples	1118
BIRT Report Designer	1121
Understanding additional report components	1122
Packaging reports for ElectricFlow	1122
Helper functions provided in ElectricCommander::ReportUtils.pm	1124
Custom Report Examples	1126
Example 1: modifying an existing report - adding a "banner" heading	1126
To copy an existing ElectricFlow report	1126
To modify the copied report design	1127
Test your new report	1129
Example 2: complete end-to-end example	1130
Planning this report example	1130

Creating a new BIRT rptdesign file	1131
Deploy your custom report	1141
Test your new report	1142
System Health Monitoring	1144
Workflow Overview	1144
Some benefits of defining a workflow include... ..	1144
Basic workflow concepts	1145
Workflow objects	1145
Workflow Definition	1146
State Definition	1146
Transition Definition	1146
Workflow	1147
State	1147
Transition	1148
Access Control	1148
Property Search Paths	1148
Parameters	1149
Sending Notifications	1150
Workflow Logs	1150
Visualizing a Workflow	1150
Building a Workflow - a Tutorial	1150
Best Practice tip	1151
A summary of the component sections to build our workflow:	1151
To begin...Calling a job from within a workflow	1152
Collecting a parameter when a workflow is launched and passing its value to a job	1153
Retrying a state	1154
Automatically transitioning to different states based on the outcome of a job	1155
Invoking another workflow	1158
Running jobs in parallel	1159
Automatically transitioning to different states based on the outcome of jobs in another workflow ..	1160

Waiting for manual intervention	1163
Restricting who can take a manual transition	1167
Sending email notifiers	1170
Adding a global parameter to use later in the workflow	1172
Setting the name of your workflow	1174
Workflow List	1176
Another workflow example	1178
Workspaces and Disk Space Management	1178
Overview	1179
Defining Workspaces	1179
Using Workspaces	1180
Workspace Directory Names	1180
Working Directories	1181
Workspace Accessibility	1181
Local Workspaces (Disconnected Workspaces)	1181
Access control	1182
Impersonation and workspaces	1182
ElectricFlow managed files	1183
Disk space management	1184
Tutorials	1184
Adding a Link to a Job	1184
Implementation	1185
Related Information	1186
Calling a Subprocedure	1186
Implementation	1186
Related Information	1187
Checking the Outcome of Preceding Steps	1187
Implementation	1187
Related information	1188
Conditional Execution	1188
Implementation	1188

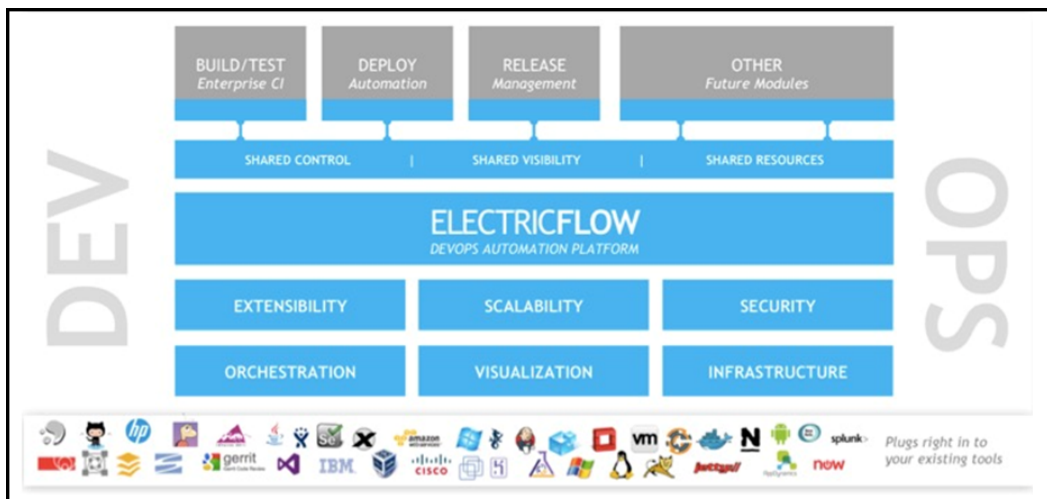
Related information	1189
Custom parameter layouts	1189
Implementation	1191
Related information	1191
Email Notifications	1191
Implementation	1192
Related information	1193
Executing Tasks on All Resources in a Pool	1193
Implementation	1193
Related information	1194
Factory Procedures	1194
Implementation	1195
Postp extension	1196
Implementation	1197
Related information	1197
Publishing and Retrieving an Artifact	1198
Implementation	1199
Related information	1199
Reserving a Resource for Job Step Duration	1199
Implementation	1200
Related information	1200
Running Steps in Parallel	1200
Implementation	1201
Related Information	1202
Step Timeouts and Steps that Always Run	1202
Implementation	1203
Storing and Retrieving Properties in a Job	1203
Implementation	1204
Related information	1204
Working with Properties Stored in a Procedure	1204

Implementation	1205
Related Information	1206

Introduction to ElectricFlow

ElectricFlow™ (including the ElectricFlow platform, formerly known as ElectricCommander) is an end-to-end Continuous Delivery application suite. It accelerates the continuous delivery of software and makes software delivery processes more repeatable, visible, scalable, and efficient. It provides domain-specific capabilities to automate some or all phases of your software delivery process, including the build, test, integrate, deploy, and release processes.

ElectricFlow gives distributed DevOps teams shared control and visibility into infrastructure, tool chains, and processes. It accelerates and automates the software delivery process and enables agility, availability, predictability, and security across many build-test, deployment, and release pipelines.



Web-Based System

At its core, ElectricFlow automation platform is a web-based system for automating and managing the build, test, deployment, and release process. It provides a scalable solution, solving some of the biggest challenges of managing these "back end" software development tasks, including these challenges:

- Time wasted on script-intensive, manual, home-grown systems that
 - Are error prone
 - Do not scale well
 - Have little or no management visibility or reporting
- Multiple, disconnected build and test systems across locations, resulting in:
 - Redundant work

- Inability to share or reuse code files across teams
- Hard to manage build and test data
- Slow overall build and release cycles that directly impact:
 - Release predictability
 - Time-to-market

Automation Platform

The automation platform has a three-tier architecture, AJAX-powered web interface, and first-of-its-kind build and release analytic capabilities for reporting and compliance. With this solution, your developers, release engineers, build managers, QA teams, and managers gain:

- Shared platform for disseminating best practices and reusing common procedures
- Ability to support geographically distributed teams
- Continuous integration and greater agility
- Faster throughput and more efficient hardware utilization
- Visibility and reporting for better project predictability
- Better software quality by integrating and validating against all target platforms and configurations

For examples of ElectricFlow architecture configurations, see [ElectricFlow Architecture](#) on page 41.

What Makes ElectricFlow Unique?

ElectricFlow provides enterprise-class speed and scalability for software build and release management. It is easy to install and use on a simple build, yet scales to support the largest and most complex build and test processes. ElectricFlow distributes jobs in parallel across multiple resources for faster overall cycle time.

ElectricFlow supports multiple teams, working in multiple locations, programming in multiple languages in an environment that can be centrally controlled and managed. Shared assets and reuse make individual teams more efficient by eliminating duplicate work, and gives organizations the power to deploy cross-company standards.

ElectricFlow's unique analytics provide visibility into one of the best indicators of project success: compiled, tested, working code. ElectricFlow's analytics database stores all build and test information for real-time and trend reporting giving your organization the power to collect pinpoint statistics and to gain visibility into important productivity metrics such as trends in error rates. Additionally, out-of-the-box reports provide information about cross-project status and build trends by project and resource utilization. ElectricFlow's integration with virtual lab automation (VLA) solutions also allows you to snapshot or reproduce a specific build for auditing or troubleshooting purposes.

ElectricFlow provides unified process automation across the entire build-test-deploy life cycle and across heterogeneous tools via integrations with leading ALM tools. Integrations with SCM tools enable continuous integration, triggering builds whenever code is checked into the specified repository/branch. When used with VMware Lab Manager, ElectricFlow can dynamically provision either physical or virtual resources without

manual intervention. This feature delivers efficient, dynamic resource provisioning and reduces development and QA dependence on IT operations.

ElectricFlow Architecture

ElectricFlow was designed to support small, mid-range, or enterprise scale software production. Based on a three-tier architecture, ElectricFlow scales to handle complex environments. The ElectricFlow multi-threaded Java server provides efficient synchronization even under high job volume.

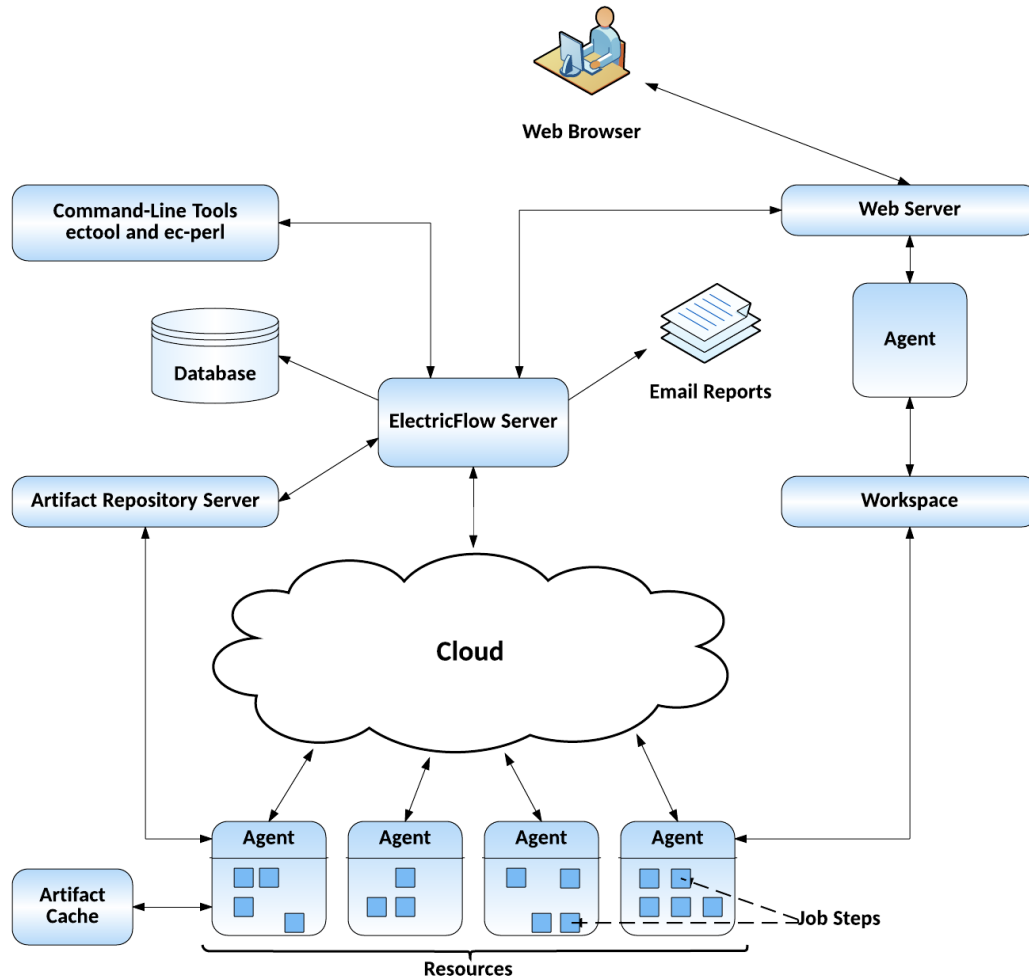
- The ElectricFlow server manages resources, issues commands, and generates reports.
- An underlying database stores commands, metadata, and log files.
- Agents execute commands, monitor status, and collect results in parallel across a cluster of servers for rapid throughput.

Simple Architectural Overview

This local configuration applies to all the use cases. The ElectricFlow server, web server, artifact cache, Artifact Repository server, workspace, command-line tools, resources, agents, and job steps are all in the automation platform.

In this local configuration:

- The ElectricFlow server manages resources, issues commands, and generates reports.
- Resources, agents, and databases are managed in the automation platform.
- An underlying database stores commands, metadata, and log files.
- Procedures, which include job steps, are defined in the automation platform.
- Job steps are executed on resources in the defined environments.
- Applications (which include procedures), components, and environments are defined for deployment automation.
- Pipelines, stages, and tasks are defined for pipeline automation.



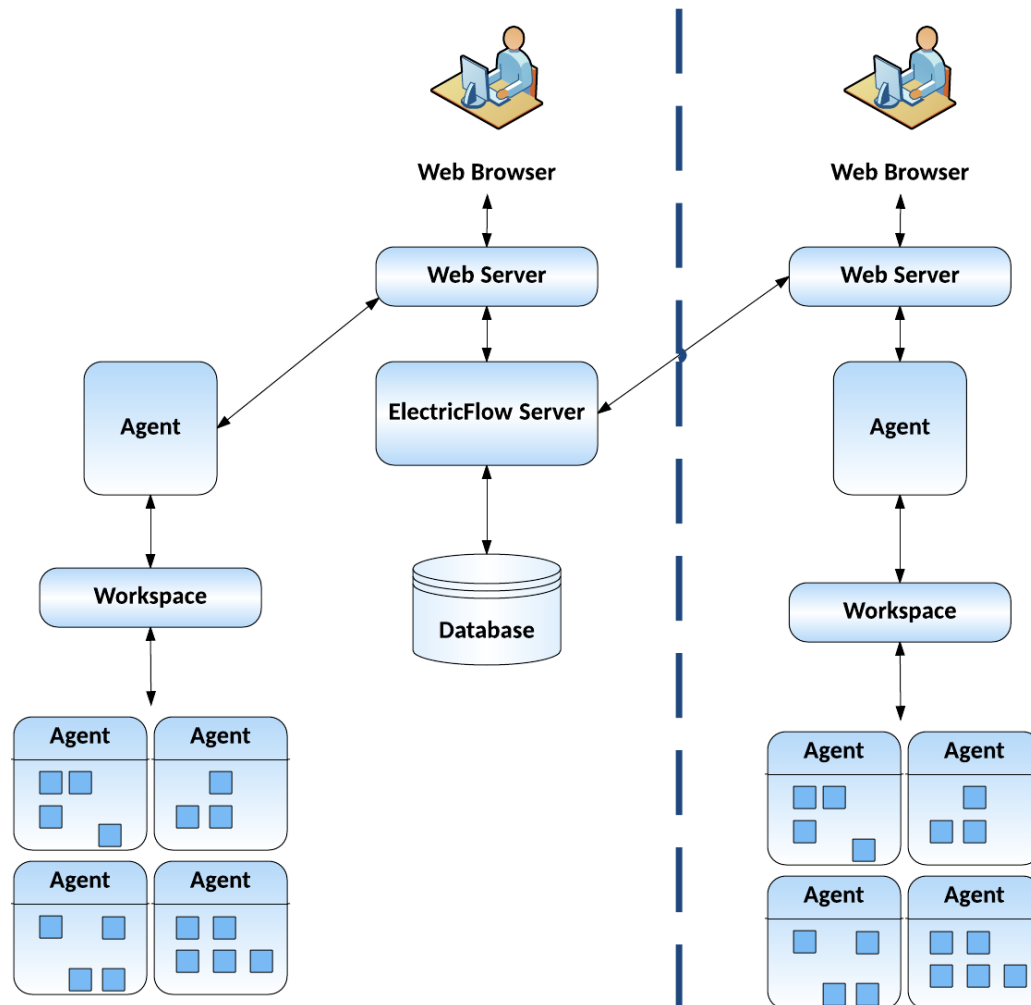
If you are only evaluating ElectricFlow, the ElectricFlow software, the database, the ElectricFlow server, the web server, and the repository server can reside on the same machine.

In a production environment, the database should reside on a separate machine from the ElectricFlow server to prevent performance issues. It is acceptable for the ElectricFlow server, web server, and repository server to reside on the same machine in a local configuration, but not required.

Expanded Remote Configuration

ElectricFlow is not limited by only the components shown in the previous configuration. This configuration applies to all the use cases.

The following shows a remote web server configuration and is an example for how you may set up a remote web server installation.



This type of remote web server configuration helps prevent network latency. If you have multiple sites, ElectricFlow can be configured to help you work more efficiently.

Other Configurations

- Proxy (universal) resources
- Remote database
- Multiple remote web servers
- Multiple remote repository servers
- Configurations designed specifically for *failover*

Build-Test Automation

You create, configure, and manage these objects in the automation platform:

For build-test automation, you must create, configure, and manage these objects in the automation platform:

- Projects

A *project* is an object used in ElectricFlow to organize information. A project is a container object for procedures, steps, schedules, workflows, and properties. If you use ElectricFlow for different purposes, you can use a separate project for each purpose so different projects do not interfere with each other. When you work in one project, you do not normally see information in other projects. At the same time, a project can use information defined in other projects, which allows you to create shared library projects.

- Resources

A *resource* is a server machine available to ElectricFlow for running steps. A resource has a logical name and a host name. In some situations, it is convenient to have multiple logical resources associated with the same host. A resource can also be associated with one or more pools. Each resource has a *step limit* that determines the maximum number of steps that can execute simultaneously on the resource. Resources can be grouped into *resource pools*.

- Procedures

Procedures and *steps* define tasks that you want ElectricFlow to execute. A procedure consists of one or more steps. A step includes a command or script executed on a single resource and is the smallest unit of work that ElectricFlow understands. Each step specifies a resource on which it should run (either the name of a specific machine or the name of a *resource pool* of equivalent machines, in which case ElectricFlow picks a machine from the pool). A step can be given a time limit, and if the step does not complete within the specified time, ElectricFlow automatically aborts it.

Steps are ordered within a procedure and normally execute sequentially. However, it is possible to mark a consecutive range of steps for parallel execution, so all steps in that range run concurrently.

You can define *parameters* for procedures. Parameter values are assigned when procedures are scheduled. Parameters can be required, optional, or have default values. Parameters are used for a variety of purposes such as specifying the branch to build or the set of platforms on which to run tests. Parameter values can be used in step commands and many other places.

Procedures can be nested. A step in one procedure can invoke another procedure instead of running a command. The invoking step provides parameters needed by the nested procedure, also referred to as a subprocedure.

- Schedules

A schedule is used to execute procedures and determine when specific procedures run. A schedule can trigger at defined times, for example, every 2 hours from 10:00 pm to 6:00 am on Mondays, Wednesdays, and Fridays, or when modifications are checked into a particular branch of your source code control system. It is also possible to create a schedule that runs immediately and disappears after the job runs. When you create a schedule, you must provide the parameters required by the procedure that you want to invoke.

The Continuous Integration Dashboard works with your source code management (SCM) system and provides visibility into running builds, the ability to add a project to continuous integration quickly, and easily accessed configuration pages to setup or modify a continuous integration schedule.

- Workflows

Managing a build-test-deploy product life cycle spanning multiple procedures and projects requires a significant amount of "meta-programming" and a heavy use of properties, and the *workflow* feature simplifies this process. Using the workflow object, you can create build-test-deploy life cycles by defining a set of states and transitions. Any ElectricFlow project can contain a workflow.

- When a procedure is executed or run, a *job* is created. A job is an object that is created each time a procedure begins to execute or run. The job keeps track of all data associated with the procedure's execution, such as the running time of each step and any errors that may occur during the step. ElectricFlow retains job information after the job completes so you can examine what occurred.

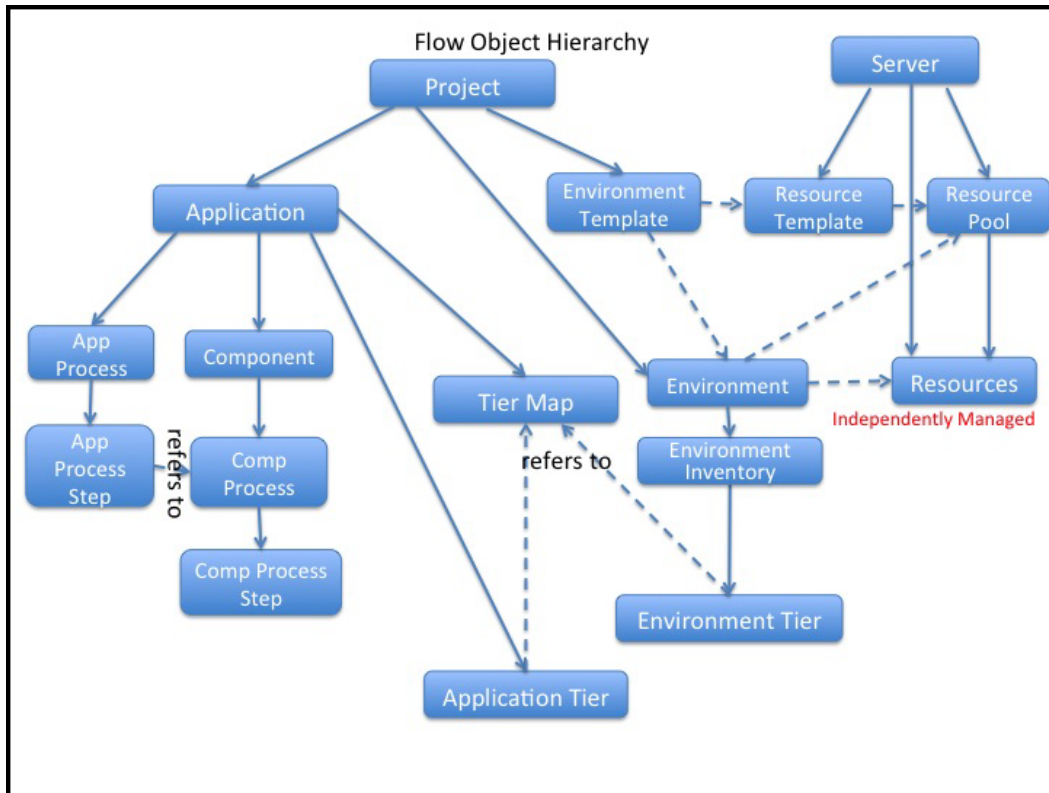
After setting resources, procedures, and schedules, ElectricFlow automatically runs the procedures that you created using these objects and facilities:

- **Zones and Gateways**—A zone (or top-level network) that you create is a way to partition a collection of agents to secure them from use by other groups. A gateway is a secured connection between two zones when you want to share or transfer information between the zones. For example, you might want a developers zone and a test zone. The ElectricFlow server is a member of the *default* zone, created during ElectricFlow installation.
- **Continuous Integration Builds** and other schedules—Run jobs according to *schedules* that you define. Scheduled jobs can run at specific times or when source code changes are checked in to your source control system. ElectricFlow integrates with major source control systems. The Continuous Integration Dashboard allows you to add more projects easily and create build configurations quickly so you can visually see running builds, build status, and so on.
- **Artifact Management** functionality—Using artifacts can improve performance across builds, provide better reusability of components, and improve cross-team collaboration with greater tractability. For example, instead of developers repeatedly downloading third-party packages from external sources, these components can be published and versioned as an artifact. Developers then simply retrieve a specific artifact version from a local repository, guaranteeing a consistent package from build to build.
- **Preflight build functionality**—Used by developers to build and test code changes in isolation on their local machines before those changes are committed to a production build.
- **Plugin** capability—ElectricFlow is built with an extensible UI, enabling easy development of plugins that include integrations with other tools, custom dashboards, and unique user experiences based on roles. "Bundled" plugins, installed during ElectricFlow installation, provide easy integration with your SCM systems, defect tracking applications, and so on.
- **Workflow** functionality—Use a workflow to design and manage processes at a higher level than individual jobs. You can use workflows to combine procedures into processes to create build-test-deploy life cycles (for example). A workflow contains states and transitions that you define to provide complete control over your workflow process. The ElectricFlow Workflow feature allows you to define an unlimited range of large or small life cycle combinations to meet your needs.
- **Resource** management—If a resource is overcommitted, ElectricFlow delays some jobs until others are finished with the resource. You can define pools of equivalent resources, and ElectricFlow spreads usage across the pool.
- Recording a variety of information about each job, such as the running time and the success or failure of each step. A set of reports is available to provide even more information.

- Powerful and flexible *reporting* facilities—Various statistics such as number of compiles or test errors are collected after each step and recorded in the ElectricFlow database. A variety of reports can be generated from this information.
- Allowing you to observe jobs as they run and to cancel jobs or change their priorities.
- *Workspace* for each job, which is a disk area a job uses for storage—ElectricFlow also provides a facility for reclaiming space occupied by workspaces.
- Powerful data model based on *properties*—Properties are used to store job input data such as the source code branch to use for the build, to collect data during a job (such as number of errors or warnings), and to annotate the job after it completes (for example, a build has passed QA).
- *Access control* for users logged into the system—ElectricFlow uses this information to control their activities and integrates with Active Directory and LDAP repositories.
- *Search, sort, and filter* functions to minimize viewing or "wading" through information that is of no interest to you, allowing you to access to the information you need quickly.
- *Email notifications* to get important information or data to individuals or groups immediately and on a regular basis for a particular job or a specific job aspect.
- All ElectricFlow operations and features are available from a command-line application tool (Perl API), *ectool*, the RESTful API, DSL methods, and a user interface (UI).

This diagram shows the relationships between objects in the automation platform to objects used in deployment automation.

- Resources are assigned to environments.
- Resource pools are also assigned to environments.
- Release pools are assigned to Resource Templates, which are used to define Environment Templates.



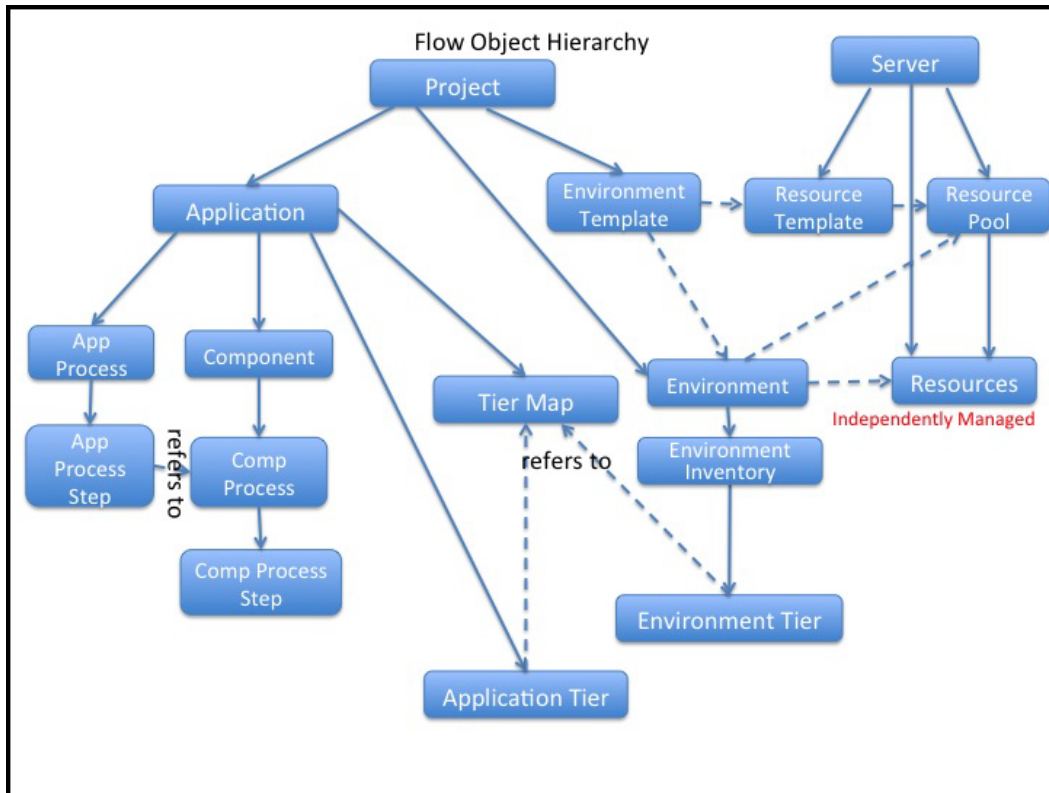
For more information about the ElectricFlow objects, concepts, and features in this topic, go to the [ElectricFlow Glossary](#) on page 53.

To configure and manage build-test automation, go to the ElectricFlow UI.

You can also use Perl API commands through `ec-perl` and `ectool`, RESTful API commands, and DSL scripts. For information about using API commands and DSL scripts, see the [ElectricFlow API Guide](#).

Deployment Automation

This diagram shows the relationships between the following objects to other objects in deployment automation.



It also shows the relationships to following objects in the automation platform:

- Resources are assigned to environments.
- Resource pools are also assigned to environments.
- Release pools are assigned to Resource Templates, which are used to define Environment Templates.

To automate your deployments for Continuous Delivery, you model and deploy (run) applications in ElectricFlow.

- *Applications* consist of application processes and application tiers.
You add components to application tiers and model component processes.
Components are based on artifacts that are defined and managed by the automation platform.
- Before deploying an application, you map an application process to an environment, where the application will be deployed, in a *tier map*.
A tier map can have one or more mappings of an application tier to an environment tier.
An environment tier can be mapped to more than one application tier.

- *Environments* can be static or dynamic.

You can create a *static environment* before deploying an application, or you can create a *dynamic environment* when deploying the application.

An environment consists of one or more environment tiers to which resources are added.

In a static environment, you can add only static resources to the environment tiers. These resources are defined and managed in the automation platform.

You can create dynamic environments with provisioned cloud resources and static resources in ElectricFlow 5.4 or later.

Apply these features in your application:

- Dynamic environments

A dynamic environment is automatically spun up on an on-demand basis when you deploy an application. It can have provisioned cloud resources and static resources.

Dynamic environments allow you to optimize how your cloud resources are used, re-use provisioned resource pools, track the status and usage of cloud resources, and verify the credentials of these resources before provisioning them.

- Deploying applications

You can deploy part or all of the objects one of these ways:

- Full deploy—All objects in the application are deployed.
- Smart deploy—Only objects that have not been deployed to specific resources, not deployed with specific artifact versions, or on new resources
- Partial deploy—Only specific objects and versions
- Schedule—On a one-time, daily, weekly, or monthly basis.
- Snapshot—Based on a version of the application with specific artifact versions and the state of the application at any point in time.

While developing an application, you can save different versions of the application as snapshots and compare them to refine and troubleshoot the application.

- Change Tracking

ElectricFlow monitors changes to *tracked* objects, such as applications, procedures, workflows, workspaces, resources, and project-owned components (such as libraries). It records a *change history* of the historical states of the system and the state changes.

- Snapshots

You can design and save a version of your application with specific artifact versions. If you save snapshots of the application during development and test phases, you can ensure that the components that were developed and tested are the same as those in the released version of the application. You can redeploy the snapshot any time.

- Credentials and impersonation

You apply credentials and impersonation to control who can run applications and where the applications are run.

- You can attach one or more credentials to component or application process steps.
- You can attach only one impersonation credential to an application process, component process, or a process step.
- When you attach an impersonation credential in ElectricFlow, it specifies the user who can deploy the application and the environment in which the application is deployed.
- When you attach an impersonation credential in the automation platform, it specifies the account (user) that can run the job or job step. If you want to specify another condition, you have to attach another credential to the object.

- Custom parameters in application processes

You can define and apply custom parameters to application processes in your deployments.

You define the parameters and apply them while deploying the application or while defining an application process step, which determines when and how the application is deployed.

- Email notifications

You can easily customize the email notification that the system sends when an application, application process, or process step runs.

When setting the recipients of email notifications, you can specify users or groups, which are defined and managed in the automation platform, as well as email addresses.

- Tracking, viewing, and troubleshooting the deployment results

Use the Environment Inventory to track and view details of the objects that were deployed and artifacts in the application. It shows the status of the application deployment at a point in time.

Use the Application Inventory to track and view the deployment results. It shows more details about the application at a point in time.

You can also view the change history of the objects in the application and search for specific information.

More about application, deploy, and run:

As you use ElectricFlow, remember that these terms have different meanings within ElectricFlow *and* outside of ElectricFlow when you deploy your software or application:

Term	Within ElectricFlow	Outside of ElectricFlow
Application	The application that you design and run (deploy) to produce your software for continuous delivery across different pipelines.	The software, system or application that you build, test, install, implement, release, and deploy using ElectricFlow. This is the end product of using ElectricFlow.
Deploy	Running the application that you designed in ElectricFlow. The end product is your software, system, or application. Deploy is a synonym of run in ElectricFlow.	All the processes or actions to develop and run your software in its environment, including building, testing, implementing, installing, configuring, making changes, and releasing.
Run	Running the application that you designed. The end product is your software, system, or application. <i>Run</i> is a synonym of <i>deploy</i> in ElectricFlow.	All the processes or actions to use software in its environment, including implementing, installing, configuring, debugging, troubleshooting, and releasing.

For more information about the ElectricFlow objects, concepts, and features in this topic, go to [ElectricFlow Glossary](#) on page 53.

To configure and manage deployment automation, go to the ElectricFlow UI.

You can also use Perl API commands through ec-perl and ectool, RESTful API commands, and DSL scripts. For information about using API commands and DSL scripts, see the [ElectricFlow API Guide](#).

Pipeline Automation

Coming soon ...

For more information about the ElectricFlow objects, concepts, and features in this topic, go to [ElectricFlow Glossary](#) on page 53.

To configure and manage pipeline automation, go to the ElectricFlow UI.

You can also use Perl API commands through ec-perl and ectool, RESTful API commands, and DSL scripts. For information about using API commands and DSL scripts, see the [ElectricFlow API Guide](#).

Roadmap to ElectricFlow

When you decide what you want to do in ElectricFlow, go to the appropriate section.

Use Case	Definition	Go to
Build-test automation	Executing automated build and test processes, resulting in reduced costs, increase quality, reliability and traceability, and accelerated time to market. Build-test automation is a key enabler of continuous integration (CI).	Build-Test Automation on page 77
Deployment automation	Executing automated application deployment processes, resulting in reduced costs, increase quality, reliability and traceability, and accelerated time to market. Deployment automation is a key enabler of Application Release Automation (ARA) and continuous delivery (CD).	Deployment Automation on page 81
Pipeline automation	Executing automated entire end-to-end software development and delivery processes, resulting in reduced costs, increase quality, reliability and traceability, and accelerated time to market. Pipeline automation is a key enabler of Application Release Automation (ARA) and Continuous Delivery (CD).	Pipeline Automation on page 525
Automation platform basics	Where you create, configure, and manage the objects that make the build-test, deployment, and pipeline automation possible.	Automation Platform Basics on page 526

ElectricFlow Glossary

These objects and concepts apply to ElectricFlow.

Object	Description
access control	An access control list (ACL) determines if a particular user can perform a particular operation on a specified object. The list contains <i>access control entries</i> (ACE), each of which specifies a user or group and indicates whether certain operations are allowed or denied for that user or group. Using access control provides security for ElectricFlow system use. See the Access Control topic for more information.
access control entry	An access control list (ACL) determines if a particular user can perform a particular operation on a specified object. The list contains <i>access control entries</i> (ACE), each of which specifies a user or group and indicates whether certain operations are allowed or denied for that user or group. Using access control provides security for ElectricFlow system use. See the Access Control topic for more information.
access control list	An access control list (ACL) determines if a particular user can perform a particular operation on a specified object. The list contains <i>access control entries</i> (ACE), each of which specifies a user or group and indicates whether certain operations are allowed or denied for that user or group. Using access control provides security for ElectricFlow system use. See the Access Control topic for more information.
ACE	See access control entry on page 53.
ACL	See access control list on page 53.
actual parameter	An actual parameter is an object that provides the value for a parameter, which is passed to a procedure when it is invoked. Actual parameters can be provided for jobs and nested subprocedures within a job. Actual parameters are different from "formal parameters": formal parameters define parameters a procedure is expecting, and actual parameters provide values to use at run-time.
admin	"admin" is a special built-in user that has universal ElectricFlow access. If you log in as admin, you can perform any operation in the system, regardless of access control limitations.

Object	Description
agent	<p>An agent is an ElectricFlow component that runs on each machine where job steps can execute. The agent works under the ElectricFlow server's control to execute job steps, monitor their progress, and record information about their completion. A single agent process can manage multiple job steps executing concurrently on a single machine.</p> <p>Note: When the ElectricFlow server and agents are installed on different hosts, make sure that the configuration for each agent specifies the Domain Name System (DNS) server.</p> <p>See the Web Interface Help > Resources topic for more information.</p>
API	<p>Application Programming Interface.</p> <p>ElectricFlow supports these APIs based on the higher-order to the lower-order programming constructs:</p> <ul style="list-style-type: none"> • DSL methods—You create scripts and templates without using API commands to execute the script. • RESTful APIs—You go to the RESTful API URI and enter the appropriate information on the API UI to execute to request. • Perl APIs—You access the command-line interface through ectool or ec-perl and enter the correct command syntax to execute the command.
application	<p><i>Application</i> can refer to one of these entities:</p> <ul style="list-style-type: none"> • Within ElectricFlow, you model and deploy (run) an application to build, test, deploy, and release your software for continuous delivery across different pipelines. An application consists of application tiers, components in these tiers, component processes, and application processes. Before running an application in ElectricFlow, you must map its application tiers to corresponding environment tiers where the application will run. • Outside of ElectricFlow, application refers to the software or application that you want to deploy, where the deployment includes build, test, installation, implementation, deploy, and release processes. This is the end product of using ElectricFlow.
Application Inventory	<p>Where ElectricFlow shows the status of an application as it runs and the results of previous runs. It includes when the application ran, how long it took, whether the application ran successfully or not, error messages, and links to more information for troubleshooting.</p>

Object	Description
application process	A group of steps, actions, or component processes taken when the ElectricFlow application is deployed. You can re-use and rerun the process more than once.
application tier	A logical grouping of components in an ElectricFlow application, which must have one or more tiers with components. A tier can have one or more components.
artifact	An artifact is a top-level object containing artifact versions, a name template for published artifact versions, artifact specific properties, and access control entries to specify privileges.
artifact key	An artifact key is an identifier for an artifact and the "key" component of the artifact name.
artifact repository	See the Artifact Management > Artifact objects > Repository topic for more information.
artifact version	An artifact version is a collection of 0 to N files that were published to an artifact repository.
automation platform	See ElectricFlow platform on page 59.
backingstore	The backingstore is the directory on the repository server where artifact versions are stored. By default, the backingstore is the <code><datadir>/repository-data</code> directory in the repository installation—this default setting can be changed.
build-test automation	Executing automated build and test processes, resulting in reduced costs, increase quality, reliability and traceability, and accelerated time to market. Build-test automation is a key enabler of continuous integration (CI).
change history	A record of the historical states of the system and the changes between them over time.
Change Tracking	How ElectricFlow monitors the changes to tracked objects, including applications, procedures, workflows, workspaces, resources, and project-owned components (such as libraries), and how it records a <i>change history</i>

Object	Description
component	<p>An object that is based on a specific version of an artifact and is defined in an application. Artifacts are defined and managed in the ElectricFlow automation platform.</p> <p>A component is the result of running an ElectricFlow application and has details, properties, and access control settings.</p> <p>It can be used by other ElectricFlow applications, or it can be the part of the continuous delivery solution. You add a component to an application tier.</p>
component process	A group of steps or actions taken on a component when the ElectricFlow application is deployed. You can re-use and rerun these processes more than once.
compression	Compression reduces transfer time when publishing an artifact. However, compression also adds overhead when computing the compressed data. If files included in the artifact version are primarily text files or are another highly compressible file format, the benefit of reduced transfer time outweighs the cost of computing compressed data.
Continuous Delivery	<p>Continuous Delivery (CD) is a set of practices and methodologies in software development designed to improve the process of software delivery and ensuring reliable software releases.</p> <p>The goal of Continuous Delivery is to keep software release-ready, and enable a repeatable, reliable way to deploy software to any environment.</p>
Continuous Integration	<p>Using continuous integration means a build is launched every time code changes are checked into a Source Control Management (SCM) system.</p> <p>The ElectricSentry component is the engine for continuous integration, while the CI Continuous Integration Dashboard is the front-end user interface for ElectricSentry.</p> <p>The Continuous Integration Manager (CI Manager) provides a front-end user interface (dashboard) for creating, managing, and monitoring continuous integration builds. For example, using your preferred SCM, you can run a procedure to build your software every time you check in code.</p>
credential	A credential is an object that stores a user name and password for later use. You can use credentials for user impersonation and saving passwords for use inside steps. Two credential types are available: <i>stored</i> or <i>dynamic</i> .

Object	Description
custom property	<p>Custom properties are identical to intrinsic properties and when placed on the same object, are referenced in the same manner and behave in every way like an intrinsic object-level property with one exception: they are not created automatically when the object is created. Instead, custom properties can be added to objects already in the database before a job is started, or created dynamically by procedure steps during step execution.</p> <p>Custom properties in a property sheet can be one of two types: <i>string</i> property or a <i>property sheet</i> property. String properties hold simple text values. Property sheet properties hold nested properties. Nested properties are accessed by way of the property sheet property of their containing sheet.</p>
deploy	<p><i>Deploy</i> can refer to one of these activities:</p> <ul style="list-style-type: none"> • Within ElectricFlow—Running the application that you modeled in ElectricFlow. The result is your software, application, or system, such as a WAR file, database, or configuration. <p>You model an application and an environment, define component and application processes, map application tiers to environment tiers, and run the application in the environment.</p> <p>Note: Within ElectricFlow, the terms <i>deploy</i> and <i>run</i> are synonymous. When deploying an application in ElectricFlow, you are actually running it to produce your software or application..</p> <ul style="list-style-type: none"> • Outside of ElectricFlow—Running all the processes, procedures, or actions to develop and deploy your software in the appropriate environment, where the deployment includes build, test, installation, implementation, deploy, and release phases.
deploying (running) applications	<p>You can deploy applications one of these ways:</p> <ul style="list-style-type: none"> • Full deploy—The system deploys all the objects (including application processes, components, and artifacts) in the application. • Smart deploy—The system deploys only the artifacts that have not been deployed to a resource or specific versions of the artifacts or have not been deployed to new resources. • Partial deploy—The system deploys only objects that you select. • Partial deploy with specific artifact versions—The system deploys only the artifacts with selected versions. • Schedule—Create schedules to run applications on a one-time, daily, weekly, or monthly basis. • Snapshot—Select a snapshot to deploy.

Object	Description
deployment automation	<p>Executing automated application deployment processes, resulting in reduced costs, increase quality, reliability and traceability, and accelerated time to market.</p> <p>Deployment automation is a key enabler of Application Release Automation (ARA) and continuous delivery (CD).</p>
description	<p>A description is an optional plain text or HTML description for an object. Description text is for your use, ElectricFlow does not use this information. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code></p>
diagnostic extract	<p>A diagnostic extract is a log file portion from a job step, typically describing an error or interesting condition, extracted by a postprocessor and saved for reporting. The postprocessor usually places this information in an XML file in the top-level job workspace directory, and then sets a property that contains the filename.</p> <p>The ElectricFlow <i>postp</i> postprocessor uses filenames like <code>diag-2770.xml</code>, where "2770" is the unique identifier for the step. Other postprocessors you may use can have a different filename configuration.</p>
domain specific language	<p>A domain specific language (DSL) is a programming language designed for a specific domain to accomplish specific tasks. You do not need to know how to use the API for the domain.</p> <p>The ElectricFlow DSL allows you to create scripts or templates for all the operations that you can do using the UI, the RESTful API, or the Perl API.</p>
DSL	See domain specific language on page 58.
dynamic credential	Dynamic credentials are captured when a job is created. Dynamic credentials are stored on the server temporarily until the job completes and then discarded.
dynamic environment	<p>An environment that is automatically created on an on-demand basis, when you deploy an application in ElectricFlow.</p> <ul style="list-style-type: none"> • An environment with only provisioned cloud resources. • An environment with only static resources added to an environment template. • An environment with provisioned cloud resources and static resources <p>The provisioned cloud resources are spun up when you are ready to deploy the application. The static resources that are part of your system or network (such as servers, databases, and agent machines).</p>

Object	Description
dynamic resource pools	Resource pools that are created by provisioning a resource or environment template in ElectricFlow.
dynamic resources	Resources in the cloud that you can provision before the application is deployed. You can also group these resources into dynamic cloud resource pools. You can define these resources in resource templates.
ec-perl	<i>ec-perl</i> is a small wrapper program installed as part of ElectricFlow. When the <i>ec-perl</i> wrapper runs, it sets up the environment, finds, and calls the ElectricFlow copy of Perl, passing all of its parameters to Perl. You can run <i>ec-perl</i> when using the ElectricFlow Perl API.
ectool	<i>ectool</i> is the ElectricFlow command-line application that provides control over the ElectricFlow system if you prefer using a command-line interface rather than the ElectricFlow web interface. Most functions that can be invoked through the ElectricFlow web interface can be invoked using <i>ectool</i> . You can run <i>ectool</i> when using the ElectricFlow Perl API.
ElectricAccelerator	ElectricAccelerator is a software build accelerator that dramatically reduces software build times by distributing the build over a large cluster of inexpensive servers. Using a patented dependency management system, ElectricAccelerator identifies and fixes problems in real time that would break traditional parallel builds. ElectricAccelerator plugs into existing Make-based infrastructures seamlessly and includes web-based management and reporting tools.
ElectricFlow platform	Software that provides domain-specific capabilities to automate some or all phases of your software delivery process, including the build, test, integrate, deploy, and release processes. automatically runs tasks and procedures and manages the objects in them (formerly known as <i>ElectricCommander</i>).
ElectricSentry	ElectricSentry is the ElectricFlow engine for continuous integration—integrating with numerous Source Control Management (SCM) systems. ElectricSentry is installed automatically with ElectricFlow and is contained in a ElectricFlow plugin named ECSCM and in the Electric Cloud project. Note: The CI Continuous Integration Dashboard is the front-end user interface for ElectricSentry.
email configuration	Before you can send an email notifier, you must set up an email configuration, which establishes communication between the ElectricFlow server and your mail server.

Object	Description
email notifier	After setting up the ElectricFlow server and your mail server to communicate, you can send email notifications (notifiers). You can attach email notifiers to procedures, procedure steps, and state definitions.
environment	<p>Within the ElectricFlow system, the location to which a resource is assigned and where jobs, procedures, application, or pipelines run. See also <i>dynamic environment</i> and <i>static environment</i>.</p> <p>Before running an application in ElectricFlow, you must map its application tiers to corresponding environment tiers where the application will run.</p>
Environment Inventory	<p>How ElectricFlow represents the status of an application as it runs in a specific environment at any point in time during the life cycle of your software.</p> <p>It tracks the application processes as they run and the results of previous runs, when the application ran, how long it took, the versions of the deployed software artifacts, the resources used, error messages, and links to more information for troubleshooting.</p>
environment template	A template defining an environment that can be spun up when the application is deployed. The template details include the environment name, its description, the environment tiers, and the resources assigned to the environment tiers. You can add one or more static resources to an environment tier. When adding resource templates to a tier, you can add only one resource template and then enter the number of dynamic resources to provision.
environment tier	A logical grouping of resources in an ElectricFlow environment, which must have one or more tiers with resources. A tier can have more than one resource.
event log	<p>Log information about system events.</p> <p>See log on page 64 for more information.</p>
Everyone	A special intrinsic access control group that includes all users.
filter	<p>Two filter categories:</p> <ul style="list-style-type: none"> • Intrinsic filters—these filters provide a convenient way to access certain well-defined fields for jobs. • Custom filters—these filters allow you to access a much broader range of values, including custom properties. Any values accessible through an intrinsic filter can be checked using a custom filter also (though not as conveniently).

Object	Description
formal parameter	<p>A formal parameter is an object that defines a parameter expected by a procedure, including its name, a default value, and an indication of whether the parameter is required.</p> <p>Formal parameters are different from <i>actual parameters</i>. Formal parameters define the kinds of parameters a procedure is expecting, and actual parameters provide values to use at run-time.</p>
gateway	<p>To communicate with a resource, workspace, or artifact repository server in another zone, a <i>gateway</i> must be created.</p> <p>A gateway object contains two resource (agent) machines. For example, <i>GatewayResource1</i> and <i>GatewayResource2</i> are configured to communicate with the other. One gateway resource resides in the <i>source</i> zone and the other in the <i>target</i> zone.</p> <p>A gateway is bidirectional and informs the ElectricFlow server that each gateway machine is configured to communicate with a gateway machine in another zone.</p>
group	<p>A group defines a collection of users for access control purposes. A group can be defined externally in an LDAP or Active Directory repository, or <i>locally</i> in the ElectricFlow server.</p> <p>See the ElectricFlow Help > Web Interface Help > Users and Groups > Groups topic for more information.</p>
impersonation	<p>Impersonation is a mechanism that allows a job step to execute under a particular login account (the ElectricFlow agent "impersonates" a particular user during the execution of that step).</p> <p>Impersonation is implemented using credentials.</p>
inheritance	<p>A feature of the ElectricFlow access control mechanism where access to a particular object is determined by the access control list for that object, and also by the access control lists of the object's parent and other ancestors. Each object can be configured to enable or disable inheritance from its ancestors.</p>
intrinsic property	<p>Intrinsic properties represent attributes that describe the object to which they are attached. ElectricFlow automatically provides intrinsic properties for each similar type object within ElectricFlow.</p> <p>For example:</p> <p>Every project has a <code>description</code> property that can be referenced with a non-local property path such as <code>/projects/Examples/description</code>.</p>

Object	Description
Inventory Tracking	<p>How ElectricFlow tracks what is deployed for continuous delivery.</p> <p>ElectricFlow tracks this information at the application and environment levels. The environment inventory is more comprehensive than the application inventory. For more information, see the application inventory and environment inventory.</p>
job	<p>A job is the output associated with invoking a ElectricFlow procedure. A new job is created each time you run (execute) a procedure.</p> <p>This records the results of running a procedure. You can view the results of running the procedure and each step within the procedure. Tables are on the Job Details page to view or manage your results or to use for future reference.</p>
job configuration	<p>A job configuration is an object containing all parameter and credential information needed to run a procedure. A Job Configuration section is provided as part of the automation platform Home page to make it easy for you to invoke your favorite configurations with a single mouse click. You can create job configurations in three ways:</p> <ul style="list-style-type: none"> • From the Job Details page for a previously invoked job, click Save Configuration at the top of the page. Your saved job configuration will be displayed on your Home page. • Create a job configuration from "scratch" by clicking Create in the Job Configurations section (on the Home page). In the Create Configuration pop-up menu, select the project and procedure you want to use for creating this configuration. • On the page for editing a schedule, click Save Configuration at the top of the page. Your saved configuration will be displayed on your Home page.
job name template	<p>This is the template used to determine the default name for jobs launched from the procedure. You can create a Job Name Template when you create a procedure. For example, in the Job Name Template field, you may enter:</p> <pre>\$[projectName]_[/increment /myproject/jobCounter]_[timestamp]</pre> <p>which produces a name like:</p> <pre>projectFoo_1234_20140102130321</pre> <p>You can enter any combination of elements to create procedure names more meaningful to you. For example, you could choose to include the build number and procedure name.</p>

Object	Description
jobs quick view	A Jobs Quick View section is one of the facilities provided on the automation platform Home page. This section allows you to define a category of jobs interesting to you (such as all running jobs or all jobs for a particular product version). Your Home page can display the last several jobs in each category you define.
job step	After a procedure is executed, the resulting job contains one job step for each step in the original procedure. The job step records information about the procedure step execution, such as the command executed, the resource where it executed, execution time, and error information.
job workspace	A directory (containing all files and subdirectories) allocated by ElectricFlow for a particular job. Each job workspace is allocated as the child of a workspace root directory.
launch pad	A starting point on the Home page from which you can go to the Applications List, Environment List, or automation platform to model the ElectricFlow application. You deploy this application to build, test, deploy, and release your software for continuous delivery.
local group	A group defined <i>inside</i> ElectricFlow, as opposed to a group defined in an external repository. A local group can refer to both local and remote users, whereas a group in an external repository refers to users in that repository only. See group on page 61 for more information.
local user	A user defined <i>inside</i> ElectricFlow, as opposed to a user defined in an external repository. If a user defined in an external repository has the same name as a local user, the external user is not accessible. Local users are not visible outside ElectricFlow. Electric Cloud recommends using external accounts whenever available, but you may need to create local users if you do not have a shared directory service or if you need special accounts to use for ElectricFlow only. See user on page 75 for more information.

Object	Description
log	<p>ElectricFlow provides a log for events generated anywhere in the system, including jobs and workflows.</p> <p>Note: From the Administration tab, the default view for the Event Log page is the warning (WARN) level. For workflow and job event logs, the default view from their respective pages is the information (INFO) level.</p> <ul style="list-style-type: none"> • To see <i>only</i> events for a single workflow, select the Workflows tab, then a workflow Name to go to the Workflow Details page and click the View Log link at the top of the page. • To see <i>only</i> events for a single job, select the Jobs tab, then the Job name to go to the Job Details page and click the View Log link at the top of the page. • To see <i>only</i> events for a specific object, select the Search tab to go to the Define Search page. For example, you can select the Object Type, "Log Entry", and then click the Add Intrinsic Filter link. Select the down-arrow where you see "Container" auto-populated and select "Container Type. Use the "equals" operator, then select the next down-arrow to choose an object. Click OK to start the search.
matcher	<p>A <i>matcher</i> controls the postp on page 65 postprocessor. Use matchers to extend postp with additional patterns if you find useful patterns in your log files undetected by postp. A matcher contains a pattern that matches lines in a step's log and actions to carry out if/when the pattern matches.</p>
misfire policy	<p>A misfire policy allows you to manage how a schedule resumes in cases where the normal scheduled time is interrupted.</p> <p>Available options are:</p> <ul style="list-style-type: none"> • <code>skip</code> (all misfires are ignored and the job runs at the next scheduled time). • <code>run once</code> (after one or more misfires, the job runs at the soonest time that occurs within an active region). See schedule on page 71 for more information.
parameter	<p>A property value passed into a procedure when it is invoked (at <i>run</i> time), and used by the procedure to change its behavior.</p> <p>Two types of parameters: actual parameter on page 53 and formal parameter on page 61.</p>

Object	Description
pipeline automation	Executing automated entire end-to-end software development and delivery processes, resulting in reduced costs, increase quality, reliability and traceability, and accelerated time to market. Pipeline automation is a key enabler of Application Release Automation (ARA) and Continuous Delivery (CD).
plugin	<p>An add-on program used by ElectricFlow to integrate with third-party tools, custom dashboards, and unique user experiences based on roles.</p> <p>ElectricFlow is built with an extensible architecture, allowing easy development of plugins that include integrations with other tools, custom dashboards, and unique user experiences based on roles. Numerous plugins are installed during the ElectricFlow installation, which makes them transparent to the user.</p> <p>For example, in the Scenario 1 and Scenario 2 Help topics, you will configure a source control system and build an Ant step, both of which use plugin technology.</p>
polling frequency	<p>The <i>polling frequency</i> is how often the ElectricSentry continuous integration engine is set to look for new code check-ins.</p> <p>The default is set to every 5 minutes.</p>
pool	<p>Also known as <i>resource pool</i>.</p> <p>A pool is a collection of resources.</p> <p>If a step specifies a pool name as its resource, ElectricFlow can choose any available resource within that pool.</p>
postp	<p><i>postp</i> is a postprocessor included with ElectricFlow. <i>postp</i> uses regular expression patterns to detect interesting lines in a step log.</p> <p><i>postp</i> is already configured with patterns to handle many common cases such as error messages and warnings from <i>gcc</i>, <i>gmake</i>, <i>cl</i>, <i>junit</i>, and <i>cppunit</i>, or any error message containing the string "error."</p> <p><i>postp</i> also supports several useful command-line options, and it can be extended using "matchers" to handle environment-specific errors.</p> <p>See matcher on page 64 for more information.</p>

Object	Description
postprocessor	<p>A postprocessor is a command associated with a particular procedure step. After a step executes, the postprocessor runs to analyze its results. Typically, a postprocessor scans the step log file to check for errors and warnings. Also, it records useful metrics such as the number of errors in properties on the job step, and extracts step log portions that provide useful information for reporting. ElectricFlow includes a standard postprocessor called <i>postp</i> for your use and you can "extend" <i>postp</i>.</p> <p>See matcher on page 64 for more information.</p>
preflight build	<p>A preflight build provides a way to build and test a developer's changes before those changes are committed. A "post-commit" source tree is simulated by creating a clean source snapshot and overlaying the developer's changes on top of it. These sources are then passed through the production build procedure to validate the changes work successfully. Developers are allowed to commit their changes only if the preflight build is successful. Because developer changes are built and tested in isolation, many common reasons for broken production builds are eliminated.</p>
privileges	<p>ElectricFlow supports four privilege types for access control on page 53 and security for each object:</p> <ul style="list-style-type: none"> • Read - Allows object contents to be viewed. • Modify - Allows object contents (but not its permissions) to be changed. • Execute - If an object is a procedure or it contains procedures (for example, a <i>project</i>), this privilege allows object procedures to be invoked as part of a job. For resource objects, this privilege determines who can use this resource in job steps. • Change Permissions - Allows object permissions to be modified.
procedure	<p>A procedure defines a process to automate one or more steps. A procedure is the ElectricFlow unit you execute (<i>run</i>) to carry out a process. A step in one procedure can call another procedure (in the same or different project), and this procedure then becomes known as a "subprocedure" (also known as a "nested" procedure). The step can pass arguments to the subprocedure.</p> <p>During deployment automation, you can also call a procedure in an application or component process.</p>
process branching	<p>How to run job steps in an application or component process on a conditional basis in ElectricFlow.</p>

Object	Description
process type	<p>Select one of the following parameters to configure how Inventory Tracking works on an application or component process in ElectricFlow:</p> <ul style="list-style-type: none"> • Deploy—Select this to enable Inventory Tracking. The ElectricFlow server tracks artifacts deployed to environments. This is the default. • Undeploy—Select this to configure the ElectricFlow automation platform to remove the environment inventory record after the first job step in a component process runs successfully. • Other—Select this to disable Inventory Tracking.
procedure	<p>A procedure contains a group of steps, each performing a task to do the work you define. You can define as many steps as you need for each procedure. Procedures are reusable indefinitely or can be modified whenever you choose.</p>
project	<p>A project is a top-level container for related procedures, workflows, schedules, jobs, and properties, which is used to isolate different user groups or functions, and also encapsulate shared facilities.</p> <p>ElectricFlow uses a project as a container for related procedures or any procedures you choose to group together. Your project can contain as many procedures as you decide are necessary.</p> <p>Projects have two purposes:</p> <ul style="list-style-type: none"> • Projects allow you to create separate work areas for different purposes or groups of people so they do not interfere with each other. In a small organization, you might choose to keep all work within a single project, but in a large organization, you may want to use projects to organize information and simplify management. • Projects simplify sharing. You can create library projects containing shared procedures and invoke these procedures from other projects. After creating a library project, you can easily copy it to other ElectricFlow servers to create uniform processes across your organization.
project principal	<p><i>Project principal</i> is a special user ID associated with each project. If a project name is "xyz," the project principal for that project is "project: xyz" (with an embedded space). This principal is used when procedures within the project are run, so you can create access control entries for this principal to control runtime behavior.</p>

Object	Description
property	<p>A property is a <code>name-value</code> pair associated with ElectricFlow objects to provide additional information beyond what is already built into the system. Built-in data is also accessible through the property mechanism. Two types of properties: <i>intrinsic</i> and <i>custom</i>.</p> <p>ElectricFlow provides intrinsic properties and allows you to create custom properties.</p> <p>Note: Intrinsic properties are case-sensitive. Custom properties, like all other object names in the ElectricFlow system, are case-preserving, but not case-sensitive.</p> <ul style="list-style-type: none"> Intrinsic properties These properties represent attributes that describe the object to which they are attached, and are automatically by ElectricFlow for each similar type object. For example, every project has a <code>Description</code> property that can be referenced with a non-local property path such as <code>/projects/Examples/description</code>. Custom properties Custom properties are identical to intrinsic properties and when placed on the same object, are referenced in the same manner, and behave in every way like an intrinsic object-level property with one exception: they are not created automatically when the object is created. Instead, custom properties can be added to objects already in the database before a job is started, or created dynamically by procedure steps during step execution.
property sheet	<p>A property sheet is a collection of properties that can be nested to any depth. The property value can be a string or a nested property sheet. Most objects have an associated "property sheet" that contains custom properties created by user scripts.</p>
proxy agent	<p>A proxy agent is an agent on a supported Linux or Windows platform, used to proxy commands to an otherwise unsupported agent platform. Proxy agents have limitations, such as the inability to work with plugins or communicate with ectool commands.</p>
proxy resource	<p>This resource type requires SSH keys for authentication. You can create proxy resources (agent on page 54 and target) for ElectricFlow to use on numerous other remote platforms/hosts that exist in your environment.</p>
proxy target	<p>A proxy target is an agent machine on an unsupported platform that can run commands via an SSH server.</p>

Object	Description
publisher	A publisher is the job that completes the <i>publish</i> operation for an artifact version.
quiet time	<p>An inactivity period before starting a build within a continuous integration system. This time period allows developers to make multiple, coordinated check-ins to ensure a build does not start with some of the changes only—assuming all changes are checked-in within the specified inactivity time period. This time period also gives developers an opportunity to "back-out" a change if they realize it is not correct. Using ElectricSentry, the inactivity time period can be configured globally for all projects or individually for a single project.</p>
RESTful API	<p>An API for web services based on the REST (REpresentational State Transfer), a simple stateless architecture.</p> <p>ElectricFlow has a RESTful API with POST, DELETE, GET, and PUT operations that are sent as HTTP requests. You do not need to have in-depth knowledge of the API to use RESTful API commands.</p>
report	<p>Reports display your information graphically for review or to show trends and other information.</p> <p>ElectricFlow provides multiple reports and custom report capabilities to help you manage your build environment.</p> <ul style="list-style-type: none"> • Real-time reports have filtered view of your workload in real-time. • Build reports are summary reports produced at the end of a build and attached to the job. • Batch reports are summaries of your build environment with trends over time: <ul style="list-style-type: none"> • Default Batch reports are automatically installed during ElectricFlow installation and scheduled to run daily (such as Cross Project Summary, Variant Trend, Daily Summary, Resource Summary, Resource Detail). • In Optional Batch reports, you can configure, rename, and schedule batch reports to fit your requirements (such as Category Report, Procedure Usage Report, Count Over Time Report, Multiple Series Reports). • In custom reports, you choose how to create and add at any time.

Object	Description
repository or repository server	<p>A <i>repository</i> is an object that stores artifact versions. This object primarily contains information about how to connect to a particular artifact repository. Similar to steps in a procedure, repository objects are in a user-specified order. When retrieving artifact versions, repositories are queried in this order until one containing the desired artifact version is found.</p> <p>Connection information is stored in the repository object on the ElectricFlow server.</p> <p>The artifact repository is a machine where artifact versions are stored in either uncompressed <code>tar</code> archives or compressed <code>tar-gzip</code> archives. The repository server is configured to store artifact versions in a directory referred to as the repository <i>backingstore</i>.</p> <p>By default, the backingstore is the <code><datadir>/repository-data</code> directory in the repository installation.</p>
resource	<p>An agent machine that is configured to communicate with ElectricFlow and where job steps can be executed.</p> <p>Steps can run on any resource or resource pool that you define.</p> <p>Resources can be grouped into a pool to increase the processing speed by assigning steps to less loaded members of the pool. Pools can also be used to ensure the resources you need are available only to a specific group of developers.</p> <p>ElectricFlow supports two types of resources:</p> <ul style="list-style-type: none"> • Standard—specifies a machine running the ElectricFlow agent on one of the supported agent platforms • Proxy—requires SSH keys for authentication. You can create proxy resources (agents and targets) for ElectricFlow to use on numerous other remote platforms/hosts that exist in your environment. <p>See also <i>dynamic resource</i> and <i>static resource</i>.</p> <p>During deployment automation, you define resources in environments during deployment automation.</p>
resource template	<p>A template with the required information to provision and later spin up cloud resources on an on-demand basis. You set the cloud provider and configuration management details in a resource template. In an environment template, you define environment tiers and can assign resource templates to the environment tiers. You can add, edit, or remove resource templates.</p>
RESTful API	<p>A method of communication between a Web-based client and server that employs representational state transfer (REST) constraints.</p>

Object	Description
run	<p><i>Run</i> can refer to these activities:</p> <ul style="list-style-type: none"> • Within ElectricFlow—Deploying the application that you modeled in ElectricFlow. The result is your software, application, or system, such as a WAR file, database, or configuration. <p>You model an application and an environment, define component and application processes, map application tiers to environment tiers, and run the application in the environment.</p> <p>Note: Within ElectricFlow, the terms <i>deploy</i> and <i>run</i> are synonymous. When deploying an application in ElectricFlow, you are actually running it to produce your software or application..</p> <ul style="list-style-type: none"> • Outside of ElectricFlow—Deploying all the processes, procedures, or actions to develop and deploy your software in the appropriate environment, where the deployment includes build, test, installation, implementation, deploy, and release phases.
schedule	<p>A schedule is an object used to execute procedures automatically in response to system events. For example, a schedule can specify executing a procedure at specific times on specific days. Three types of schedules are available:</p> <ul style="list-style-type: none"> • Standard—A standard schedule defines when your procedure will run. You can control the hour and day that your procedure runs or use a trigger to run a procedure each time a particular event occurs. • Continuous Integration • Custom—Custom schedules are typically continuous integration schedules that do not use the ECSCM plugin.
Sentry schedule	<p>A continuous integration schedule created using the ElectricSentry engine for continuous integration or the CI Continuous Integration Dashboard, which is an easy- to-use front-end user interface for the ElectricSentry engine.</p>
shortcut	<p>One type of shortcut is part of the automation platform Home page facility and records the location of a page you visit frequently (either inside or outside of ElectricFlow), so you can return to that page with a single click from the Home page.</p> <p>Another type of shortcut is a context-relative shortcut to property paths. This shortcut can be used to reference a property without knowing the exact name of the object that contains the property. You might think of a shortcut as another part of the property hierarchy. These shortcuts resolve to the correct property path even though its path elements may have changed because a project or procedure was renamed. Shortcuts are particularly useful if you do not know your exact location in the property hierarchical tree.</p>

Object	Description
snapshot	A version of an application with specific artifact versions and the state of the application at any point in time.
state	Workflows always have a single active <i>state</i> . Each state in a workflow, when it becomes active, can perform an action. A state can run a procedure to create a subjob or run a workflow definition to create a subworkflow—in the same way that procedures can call other procedures. One or more states can be designated as "starting" states to provide multiple entry points into the workflow.
state definition	<p>Workflow objects are split into two types: <i>Definition</i> objects and <i>Instance</i> objects. Definition objects provide the template for a running workflow instance. You create a new workflow by defining a Workflow Definition along with its State Definition and Transition Definition objects.</p> <p>When you run the workflow definition, the system creates a new workflow object with an equivalent set of State and Transition objects that represent the run-time instances of the workflow definition.</p> <p>Note: We omit the "Instance" qualifier for brevity in the API and the UI.</p> <p>Each workflow can contain one or more state objects. Defining states for a workflow is analogous to defining steps for a procedure.</p>
static environment	An environment with resources that are in your system or network, such as servers, databases, and agent machines. You model this environment and assign static resources to it before deploying the application.
static resource	A resource located in your system or network, not in the cloud. Servers, databases, and agent machines are examples of static resources.

Object	Description
step	<p>A step is part of a procedure or process. Using almost any kind of script, you can define a specific task for a step to perform. Steps can run in parallel for faster processing. Steps can be modified or copied to other procedures whenever you choose.</p> <p>Each step specifies a command to execute on a particular resource or a subprocedure (nested procedure) to invoke. Commonly created steps include:</p> <ul style="list-style-type: none"> • Command—This step invokes a <code>bat</code>, <code>cmd</code>, <code>shell</code>, <code>perl</code> script, or similar. • Subprocedure—This step invokes another ElectricFlow procedure. • Plugin step—These include task-specific steps. Depending on which step-type you choose, the information you need to enter is somewhat different. Some of the step types bundled with ElectricFlow include: <ul style="list-style-type: none"> • Publish or retrieve artifact version • Send Email • Various SCM step types • Build tools
stored credential	<p><i>Stored credentials</i> are given a name and stored in encrypted form in the database. Each project contains a list of stored credentials it owns.</p> <p>These credentials are managed from the Project Details page.</p>
subprocedure	<p>Creating subprocedures is a way of "nesting" procedures. A step (from any procedure) can call a procedure from another project or the same project. The procedure called by the step then becomes a subprocedure.</p>
substitution	<p>A mechanism used to include property values in step commands and elsewhere.</p> <p>For example, if a step command is specified as <code>echo \${status}</code>, and when the step executes there is a property named <code>status</code> with value <code>success</code>, the actual command executed will be <code>echo success</code>.</p>
system object	<p>This is a special object whose access control lists are used to control access to some ElectricFlow internals.</p> <p>System objects are: <code>admin</code>, <code>artifactVersions</code>, <code>directory</code>, <code>emailConfigs</code>, <code>forceAbort</code>, <code>licensing</code>, <code>log</code>, <code>plugins</code>, <code>priority</code>, <code>projects</code>, <code>repositories</code>, <code>resources</code>, <code>server</code>, <code>session</code>, and <code>workspaces</code>.</p>

Object	Description
tag	<p>A way to categorize a project to identify its relationship to one or more other projects or groups. You can edit a project to add a tag. Enter a tag if you want to categorize or "mark" a project to identify its relationship to one or more other projects or groups.</p> <p>For example, you might want to tag a group of projects as "production" or "workflow", or you might want to use your name so you can quickly sort the project list to see only those projects that are useful to you.</p>
tier map	<p>The mapping of the application tiers to the corresponding environment tiers where the application will run.</p> <p>To run an application, you map one application tier to one or more environment tiers and must have at least tier map.</p>
transition	<p>Transitions are used to move workflow progress from one state to another state. Four types of transitions are available to move a workflow to the next state:</p> <ul style="list-style-type: none">• On Enter - transitions before sending notifiers or starting the sub-action• On Start - transitions immediately after starting the sub-action. These transitions are ignored if no sub-action is specified for the source state.• On Completion - transitions when the sub-action completes. These transitions are ignored if no sub-action is specified for the source state. <p>Note: On Completion transitions are taken only if the state is still active when the sub-action completes, and are ignored if the workflow has transitioned to another state—this can occur if an On Start or Manual transition occurred before the sub-action completed.</p> <ul style="list-style-type: none">• Manual - transitions when a user selects the transition in the UI and specifies parameters. The same action can occur using ectool or the Perl API by calling transitionWorkflow. Only users who have "execute" permission on the transition are allowed to use the Manual transition.

Object	Description
transition definition	<p>How a workflow transitions from one state to another.</p> <p>When you run the workflow definition, the system creates a new Workflow object with an equivalent set of State and Transition objects that represent the run-time instances of the workflow definition.</p> <p>Note: We omit the "Instance" qualifier for brevity in the API and the UI.</p> <p>Each state can contain one or more transition objects. The transition definition object requires a name for the transition. This transition name will appear on the Workflow Definition Details page for quick reference and also on the State Definition Details page when you select the Transition Definitions tab.</p> <p>You can define one or more transitions for each state, depending on which transition options you want to apply to a particular state.</p>
user	<p>A user defines an account used to log into the system and control access to ElectricFlow objects. A user can be defined externally in an LDAP or Active Directory repository, or locally in ElectricFlow.</p> <p>See local user on page 63 for more information.</p>
workflow	<p>You can use a workflow to design and manage processes at a higher level than individual jobs. For example, workflows allow you to combine procedures into processes to create build-test-deploy lifecycles.</p> <p>A workflow contains <i>states</i> and <i>transitions</i> you define to provide complete control over your workflow process. The ElectricFlow Workflow feature allows you to define an unlimited range of large or small lifecycle combinations to meet your needs.</p> <p>Workflow objects are split into two types: <i>Definition</i> objects and <i>Instance</i> objects. Definition objects provide the template for a running workflow instance. You create a new workflow by defining a Workflow Definition along with its State Definition and Transition Definition objects.</p>
workflow definition	<p>Workflow objects are split into two types: <i>Definition</i> objects and <i>Instance</i> objects. Definition objects provide the template for a running workflow instance. You create a new workflow by defining a Workflow Definition along with its State Definition and Transition Definition objects.</p> <p>When you run the workflow definition, the system creates a new Workflow object with an equivalent set of State and Transition objects that represent the run-time instances of the workflow definition.</p> <p>Note: We omit the "Instance" qualifier for brevity in the API and the UI.</p>

Object	Description
workflow name template	<p>This is the template used to determine the default name of jobs launched from the workflow definition.</p> <p>For example:</p> <pre>[projectName]_[/increment /myproject/workflowCounter]_[timestamp]</pre> <p>(substitute your values for the names above)</p> <p>Produces a workflow name like:</p> <pre>projectName_123_20140102130321</pre>
workspace	<p>A workspace is a subtree of files and directories where job file data is stored. The term "workspace" typically refers to the top-level directory in this subtree.</p> <p>ElectricFlow provides a default area on the disk that it can use for working files and results. This disk area is called a job workspace.</p> <p>A job step can create whatever files it needs in its workspace, such as step logs, and ElectricFlow automatically places these files in the workspace.</p> <p>A single workspace can be shared by all steps in a job, but it is possible for different steps in a job to use different workspaces.</p> <p>The location of the job step workspace is displayed on the Job Details page for the job under the "Details" for the step.</p>
workspace root	<p>A workspace root is a directory in which ElectricFlow allocates job workspace directories.</p> <p>Each workspace root has a logical name used to refer to it in steps and procedures.</p>
zones	<p>A zone is a way to partition a collection of agents to secure them from use by other groups.</p> <p>For example, you might choose to create a developers zone, a production zone, and a test zone. The agents in one zone cannot directly communicate with agents in another zone.</p> <p>A <i>default</i> zone is created during ElectricFlow installation.</p> <p>The server implicitly belongs to the default zone, which means all agents in this zone can communicate with the server directly (without the use of a gateway).</p>

Build-Test Automation

In this use case, ElectricFlow executes build and test production processes automatically, triggered by a schedule or event (such as every time that code is checked into a source code management [SCM] system).

ElectricFlow can perform tasks such as:

- Configuring resources.
- Defining procedures that contain steps and subprocedures.
- Determining when to execute procedures based on a schedule or event.
- Configuring the credentials and impersonation settings to limit who can run a procedure or procedure step.
- Tracking the Change History of tracked objects, such as artifacts, resources, and ACLs.
- Performing continuous integration of your builds.

For more information about the ElectricFlow objects, concepts, and features in this topic, go to the [ElectricFlow Glossary](#) on page 53

You use only the automation platform UI to configure ElectricFlow to perform these tasks. The objects and operations for build-test automation are the same as those for the automation platform.

You can also use Perl API commands through `ec-perl` and `ectool`, RESTful API commands, and DSL scripts. For information about using API commands and DSL scripts, see the [ElectricFlow API Guide](#).

Getting Started on Build-Test Automation

You first need to do these tasks on the automation platform:

- Configure *resources*—These are machines ElectricFlow can use to run jobs.
- Define *procedures*—These are the tasks you want to automate and contain one or more steps.
- Create *schedules*—These define when each procedure will run.

For more information, go to [Getting Started on the Automation Platform](#).

Build-Test Automation Concepts, Features, and Functionality

For build-test automation, you must create, configure, and manage these objects in the automation platform:

- Projects

A *project* is an object used in ElectricFlow to organize information. A project is a container object for procedures, steps, schedules, workflows, and properties. If you use ElectricFlow for different purposes, you can use a separate project for each purpose so different projects do not interfere with each other. When you work in one project, you do not normally see information in other projects. At the same time, a project can use information defined in other projects, which allows you to create shared library projects.

- Resources

A *resource* is a server machine available to ElectricFlow for running steps. A resource has a logical name and a host name. In some situations, it is convenient to have multiple logical resources associated with the same host. A resource can also be associated with one or more pools. Each resource has a *step limit* that determines the maximum number of steps that can execute simultaneously on the resource. Resources can be grouped into *resource pools*.

- Procedures

Procedures and *steps* define tasks that you want ElectricFlow to execute. A procedure consists of one or more steps. A step includes a command or script executed on a single resource and is the smallest unit of work that ElectricFlow understands. Each step specifies a resource on which it should run (either the name of a specific machine or the name of a *resource pool* of equivalent machines, in which case ElectricFlow picks a machine from the pool). A step can be given a time limit, and if the step does not complete within the specified time, ElectricFlow automatically aborts it.

Steps are ordered within a procedure and normally execute sequentially. However, it is possible to mark a consecutive range of steps for parallel execution, so all steps in that range run concurrently.

You can define *parameters* for procedures. Parameter values are assigned when procedures are scheduled. Parameters can be required, optional, or have default values. Parameters are used for a variety of purposes such as specifying the branch to build or the set of platforms on which to run tests. Parameter values can be used in step commands and many other places.

Procedures can be nested. A step in one procedure can invoke another procedure instead of running a command. The invoking step provides parameters needed by the nested procedure, also referred to as a subprocedure.

- Schedules

A schedule is used to execute procedures and determine when specific procedures run. A schedule can trigger at defined times, for example, every 2 hours from 10:00 pm to 6:00 am on Mondays, Wednesdays, and Fridays, or when modifications are checked into a particular branch of your source code control system. It is also possible to create a schedule that runs immediately and disappears after the job runs. When you create a schedule, you must provide the parameters required by the procedure that you want to invoke.

The Continuous Integration Dashboard works with your source code management (SCM) system and provides visibility into running builds, the ability to add a project to continuous integration quickly, and easily accessed configuration pages to setup or modify a continuous integration schedule.

- Workflows

Managing a build-test-deploy product life cycle spanning multiple procedures and projects requires a significant amount of "meta-programming" and a heavy use of properties, and the *workflow* feature simplifies this process. Using the workflow object, you can create build-test-deploy life cycles by defining a set of states and transitions. Any ElectricFlow project can contain a workflow.

- When a procedure is executed or run, a *job* is created. A job is an object that is created each time a procedure begins to execute or run. The job keeps track of all data associated with the procedure's execution, such as the running time of each step and any errors that may occur during the step. ElectricFlow retains job information after the job completes so you can examine what occurred.

After setting resources, procedures, and schedules, ElectricFlow automatically runs the procedures that you created using these objects and facilities:

- **Zones and Gateways**—A zone (or top-level network) that you create is a way to partition a collection of agents to secure them from use by other groups. A gateway is a secured connection between two zones when you want to share or transfer information between the zones. For example, you might want a developers zone and a test zone. The ElectricFlow server is a member of the *default* zone, created during ElectricFlow installation.
- **Continuous Integration Builds** and other schedules—Run jobs according to *schedules* that you define. Scheduled jobs can run at specific times or when source code changes are checked in to your source control system. ElectricFlow integrates with major source control systems. The Continuous Integration Dashboard allows you to add more projects easily and create build configurations quickly so you can visually see running builds, build status, and so on.
- **Artifact Management** functionality—Using artifacts can improve performance across builds, provide better reusability of components, and improve cross-team collaboration with greater tractability. For example, instead of developers repeatedly downloading third-party packages from external sources, these components can be published and versioned as an artifact. Developers then simply retrieve a specific artifact version from a local repository, guaranteeing a consistent package from build to build.
- **Preflight build functionality**—Used by developers to build and test code changes in isolation on their local machines before those changes are committed to a production build.
- **Plugin** capability—ElectricFlow is built with an extensible UI, enabling easy development of plugins that include integrations with other tools, custom dashboards, and unique user experiences based on roles. "Bundled" plugins, installed during ElectricFlow installation, provide easy integration with your SCM systems, defect tracking applications, and so on.
- **Workflow** functionality—Use a workflow to design and manage processes at a higher level than individual jobs. You can use workflows to combine procedures into processes to create build-test-deploy life cycles (for example). A workflow contains states and transitions that you define to provide complete control over your workflow process. The ElectricFlow Workflow feature allows you to define an unlimited range of large or small life cycle combinations to meet your needs.
- **Resource** management—If a resource is overcommitted, ElectricFlow delays some jobs until others are finished with the resource. You can define pools of equivalent resources, and ElectricFlow spreads usage across the pool.
- Recording a variety of information about each job, such as the running time and the success or failure of each step. A set of reports is available to provide even more information.
- Powerful and flexible *reporting* facilities—Various statistics such as number of compiles or test errors are collected after each step and recorded in the ElectricFlow database. A variety of reports can be generated from this information.
- Allowing you to observe jobs as they run and to cancel jobs or change their priorities.
- **Workspace** for each job, which is a disk area a job uses for storage—ElectricFlow also provides a facility for reclaiming space occupied by workspaces.
- Powerful data model based on *properties*—Properties are used to store job input data such as the source code branch to use for the build, to collect data during a job (such as number of errors or warnings), and to annotate the job after it completes (for example, a build has passed QA).
- **Access control** for users logged into the system—ElectricFlow uses this information to control their activities and integrates with Active Directory and LDAP repositories.
- **Search, sort, and filter** functions to minimize viewing or "wading" through information that is of no interest to you, allowing you to access the information you need quickly.

- *Email notifications* to get important information or data to individuals or groups immediately and on a regular basis for a particular job or a specific job aspect.
- All ElectricFlow operations and features are available from a command-line application tool (Perl API), *ectool*, the RESTful API, DSL methods, and a user interface (UI).

Roadmap for Build-Test Automation Tasks

Information	Go to
Setup tasks	Automation Platform Setup
Build-test automation tasks	Automation Platform Tasks on page 629

Deployment Automation

In this use case, ElectricFlow deploys application processes automatically, triggered by a schedule (such as a software build every two hours) or event.

ElectricFlow can perform tasks such as:

- Deploy all or some of the artifact in an application.
- Specifying which parameters must be set to deploy an application.
- Saving a snapshot of an application at a point in time.
- Provisioning and spinning up cloud resources on an on-demand basis to model a dynamic environment.
- Making decisions about the next step in an application or component process based on the result of the previous step.
- Defining process steps based on procedures managed in the automation platform (subprocedures), plugins, or scripts.
- Creating and managing resources.
- Tracking the change history of tracked objects, such as artifacts, resources, and access control lists (ACLs).
- Performing Continuous Delivery (CD) of your software.

You use all of the ElectricFlow UI to configure ElectricFlow to perform these tasks.

For more information about the ElectricFlow objects, concepts, and features in this topic, go to [ElectricFlow Glossary](#) on page 53

You can also use Perl API commands through `ec-perl` and `ectool`, RESTful API commands, and DSL scripts. For information about using API commands and DSL scripts, see the [ElectricFlow API Guide](#).

Getting Started with Deployment Automation

You first need to do these tasks:

- Configure *resources*—These are machines ElectricFlow can use to run jobs.
- Define *procedures*—These are the tasks you want to automate and contain one or more steps.
- Create *schedules*—These define when each procedure will run.

Then you model an application consisting of application processes and component processes and deploy it. The process steps can be based on procedures or plugins.

This is a typical sequence of steps to model and deploy an application in ElectricFlow:

1. Create the application and application tiers.
2. Create components and component processes.
3. Create application processes.
4. Create resources.
5. Create environments and environment tiers, and assign resources to the tiers.

6. Map application tiers to environment tiers.
7. Deploy (run) the application.

Deployment Automation Concepts and Objects

To automate your deployments for Continuous Delivery, you model and deploy (run) applications in ElectricFlow.

- *Applications* consist of application processes and application tiers.

You add components to application tiers and model component processes.

Components are based on artifacts that are defined and managed by the automation platform.

- Before deploying an application, you map an application process to an environment, where the application will be deployed, in a *tier map*.

A tier map can have one or more mappings of an application tier to an environment tier.

An environment tier can be mapped to more than one application tier.

- *Environments* can be static or dynamic.

You can create a *static environment* before deploying an application, or you can create a *dynamic environment* when deploying the application.

An environment consists of one or more environment tiers to which resources are added.

In a static environment, you can add only static resources to the environment tiers. These resources are defined and managed in the automation platform.

You can create dynamic environments with provisioned cloud resources and static resources in ElectricFlow 5.4 or later.

Apply these features in your application:

- Dynamic environments

A dynamic environment is automatically spun up on an on-demand basis when you deploy an application. It can have provisioned cloud resources and static resources.

Dynamic environments allow you to optimize how your cloud resources are used, re-use provisioned resource pools, track the status and usage of cloud resources, and verify the credentials of these resources before provisioning them.

- Deploying applications

You can deploy part or all of the objects one of these ways:

- Full deploy—All objects in the application are deployed.
- Smart deploy—Only objects that have not been deployed to specific resources, not deployed with specific artifact versions, or on new resources
- Partial deploy—Only specific objects and versions
- Schedule—On a one-time, daily, weekly, or monthly basis.
- Snapshot—Based on a version of the application with specific artifact versions and the state of the application at any point in time.

While developing an application, you can save different versions of the application as snapshots and compare them to refine and troubleshoot the application.

- Change Tracking

ElectricFlow monitors changes to *tracked* objects, such as applications, procedures, workflows, workspaces, resources, and project-owned components (such as libraries). It records a *change history* of the historical states of the system and the state changes.

- Snapshots

You can design and save a version of your application with specific artifact versions. If you save snapshots of the application during development and test phases, you can ensure that the components that were developed and tested are the same as those in the released version of the application. You can redeploy the snapshot any time.

- Credentials and impersonation

You apply credentials and impersonation to control who can run applications and where the applications are run.

- You can attach one or more credentials to component or application process steps.
- You can attach only one impersonation credential to an application process, component process, or a process step.
- When you attach an impersonation credential in ElectricFlow, it specifies the user who can deploy the application and the environment in which the application is deployed.
- When you attach an impersonation credential in the automation platform, it specifies the account (user) that can run the job or job step. If you want to specify another condition, you have to attach another credential to the object.

- Custom parameters in application processes

You can define and apply custom parameters to application processes in your deployments.

You define the parameters and apply them while deploying the application or while defining an application process step, which determines when and how the application is deployed.

- Email notifications

You can easily customize the email notification that the system sends when an application, application process, or process step runs.

When setting the recipients of email notifications, you can specify users or groups, which are defined and managed in the automation platform, as well as email addresses.

- Tracking, viewing, and troubleshooting the deployment results

Use the Environment Inventory to track and view details of the objects that were deployed and artifacts in the application. It shows the status of the application deployment at a point in time.

Use the Application Inventory to track and view the deployment results. It shows more details about the application at a point in time.

You can also view the change history of the objects in the application and search for specific information.

More about application, deploy, and run:

As you use ElectricFlow, remember that these terms have different meanings within ElectricFlow *and* outside of ElectricFlow when you deploy your software or application:

Term	Within ElectricFlow	Outside of ElectricFlow
Application	The application that you design and run (deploy) to produce your software for continuous delivery across different pipelines.	The software, system or application that you build, test, install, implement, release, and deploy using ElectricFlow. This is the end product of using ElectricFlow.
Deploy	Running the application that you designed in ElectricFlow. The end product is your software, system, or application. Deploy is a synonym of run in ElectricFlow.	All the processes or actions to develop and run your software in its environment, including building, testing, implementing, installing, configuring, making changes, and releasing.
Run	Running the application that you designed. The end product is your software, system, or application. <i>Run</i> is a synonym of <i>deploy</i> in ElectricFlow.	All the processes or actions to use software in its environment, including implementing, installing, configuring, debugging, troubleshooting, and releasing.

What You Can Do for Deployment Automation

These are a few ways that you can use ElectricFlow to improve deployment automation.

One Application for Multiple Use Cases

In ElectricFlow, you can model an application that works for more than one deployment scenario. For example, you can deploy software releases for Linux, Javascript, and Windows using the same application. You can also specify who can run specific processes in the application and the environment in which the processes are run without modeling and running applications for each scenario.

When modeling the application and the environments in which it runs, you can implement the following:

- Process branching

Model an application or component process with more than one path. Decisions about the next step are made while the process runs.

ElectricFlow uses conditions (success, failure, or custom) to determine the path through the application or component process.

- Credentials and impersonation

You can control who runs an application or component process step and where that step runs (the environment) using credentials.

You can also attach one impersonation credential to an application process, a component process, or a process step to give a user higher order privileges for the only that part of the process or process step.

Keeping Users Up-To-Date

You can set email notifications, which are sent to users or groups who are interested in or need to know the results. Notifications are sent when the application, application process, or process step succeeds or fails. You can select recipients by specifying the user name, which is defined and managed in the automation platform, or the user's email address. It is easy to customize the text for the email notification in the ElectricFlow UI.

Keeping Track of Changes

Change Tracking is available for more reliable and repeatable software deployments. ElectricFlow tracks changes to *tracked* objects including applications, artifacts, components, application and component process steps, jobs, resources, and workflows. It records a *change history* of the historical states of the system and the state changes.

You can use Change Tracking in these scenarios:

- When you are debugging a failed job or want to more information about a component, see the change history for the changes relevant to that object.
- When you search for specific change history records, filter the records by time frame, change type, entity type, or developer.
- Revert changes to an object or to an objects and its children.
- When you want to determine the differences between objects, export them at various levels in the object hierarchy.

Taking Snapshots

Snapshots are available for more reliable and repeatable software deployments.

- You can model and save a version of your application with specific artifact versions and rerun it later, even if the latest version of the application changed.
- If you save snapshots of the application during development and test phases, you can ensure that the components that were developed and tested are the same as those in the released version of the application. You can redeploy the snapshot any time.
- You can create and save more than one snapshot for different deployment scenarios.
- You can view the snapshots in the Snapshot List. From this list, you can manage all your snapshots, compare two snapshots, or get more information about them.
- Comparing snapshots helps you to deploy applications with reliable and repeatable results during ongoing cycles of software releases. You can build and test applications using snapshots, and do not have to design a new application for each release.
- You can use snapshots to refine and optimize an application that fits your deployment scenario and ensure that this version is properly developed, tested, and released.

Optimizing How Resources are Used

Starting in ElectricFlow 5.4, you can model and create *dynamic environments* that are automatically spun up when an application is deployed. These environments can have cloud and static resources.

You provision cloud resources by using resource templates. When you are ready to deploy the application, use environment templates to dynamically create the environment.

Using dynamic environments allows you to do the following:

- Provide ways to optimize how cloud resources are used.
- Re-use provisioned resource pools.
- Track how provisioned cloud resources are used.
- Provide the status of the provisioning process.
- Verify the credentials of cloud resources before provisioning them.
- Configure the middleware of cloud resources on an on-demand basis.

Customizing Deployments

During the development, testing, and implementation of your application, you can deploy the application several ways, depending on what you want to do and where you are in the continuous delivery cycle.

- Full deploy—Deploy all of the artifacts in the application.
- Partial deploy—Deploy some of the artifacts by specifying the objects of the application to deploy or by specifying the artifact versions of the objects.
- Smart deploy—Deploy only the artifacts that have not been deployed to a resource, specific versions of artifacts, or artifacts that have not been yet deployed to new resources.
- Based on a schedule—Deploy the application a one-time, daily, weekly, or monthly basis.
- Snapshot—Select a snapshot to deploy.
- Based on custom parameters defined in application processes—Set these parameters when you deploy the application or when you define an application process step. They determine how the application should be deployed.

Getting the Real-Time Status of Application Runs and Troubleshooting

Go to the Application Inventory and the Environment Inventory to view the progress of the application as it runs and the results of previous runs. They show detailed results that can be used to troubleshoot the application.

- In the Application Inventory, you can get information about the application, its application processes, components, and job steps and about the status of these objects.
- In the Environment Inventory, you can get more details about the environment, the applications mapped to it, number of deployed artifacts in the applications, where the artifacts are deployed, and the status of these objects.

You can also create and compare snapshots.

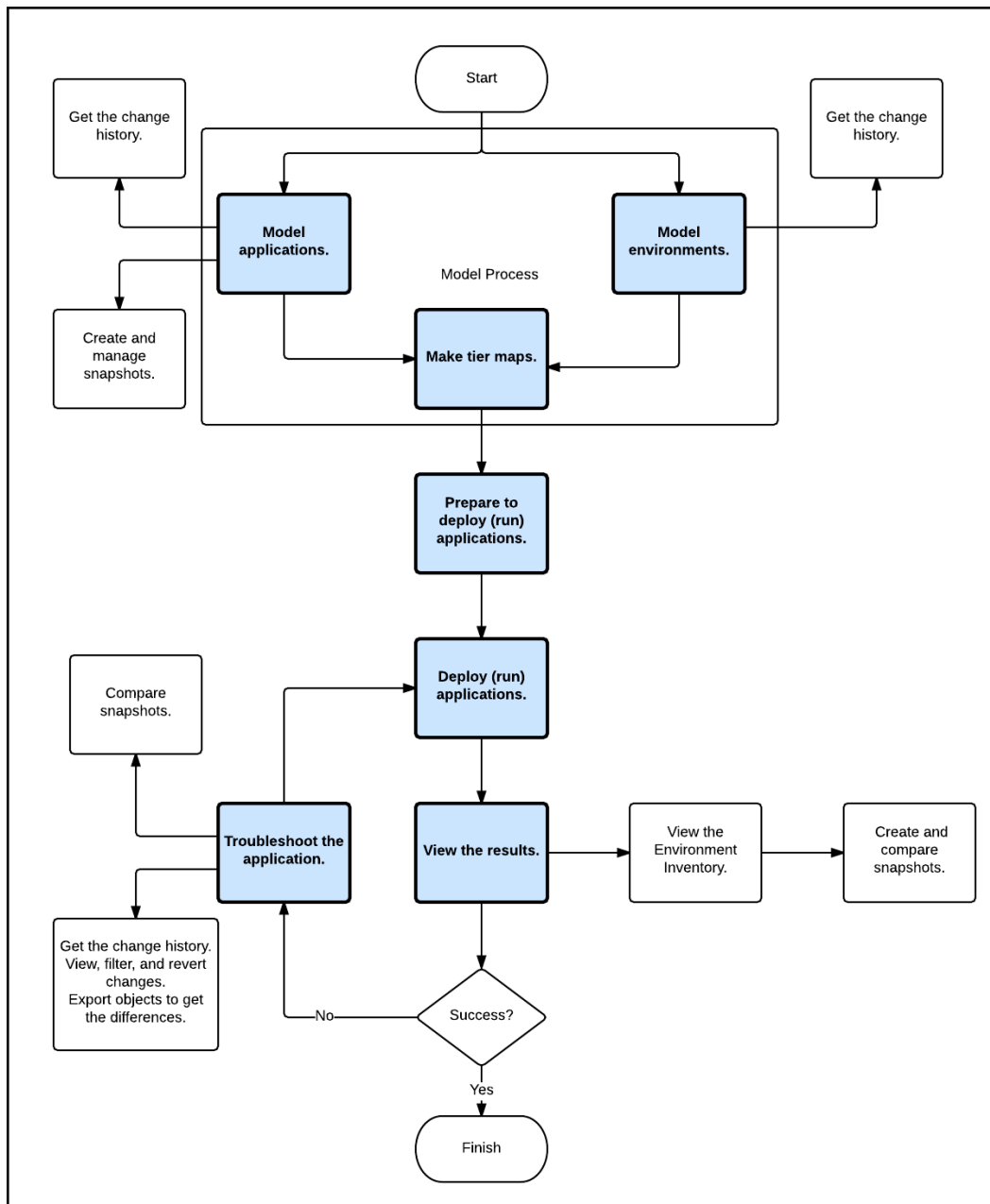
Modeling and Deploying Applications in Static Environments

This topic describes how to model an application, deploy it in a static environment, and view and troubleshoot the results. In this workflow, you create a static environment with resources in your system or network, such as servers, databases, machines, and virtual machines (VMs), before deploying the application. It does not describe how to create dynamic environments with cloud resources that are spun up when you deploy the application. For information about the dynamic environment workflow, go to [Modeling and Deploying Applications in Dynamic Environments](#) on page 191.

The following tasks describe how to model and deploy (run) applications at a high level.

1. [Logging in to ElectricFlow](#) on page 194
2. [Modeling Applications](#) on page 195
3. [Modeling Environments](#) on page 146
4. [Making Tier Maps](#) on page 150
5. [Deploying \(Running\) Applications](#) on page 152
6. [Viewing Results and Troubleshooting](#) on page 182

For information about the UI, see [Deployment Automation User Interface](#) on page 333.



More about application, deploy, and run:

As you use ElectricFlow, remember that these terms have different meanings within ElectricFlow *and* outside of ElectricFlow when you deploy your software or application:

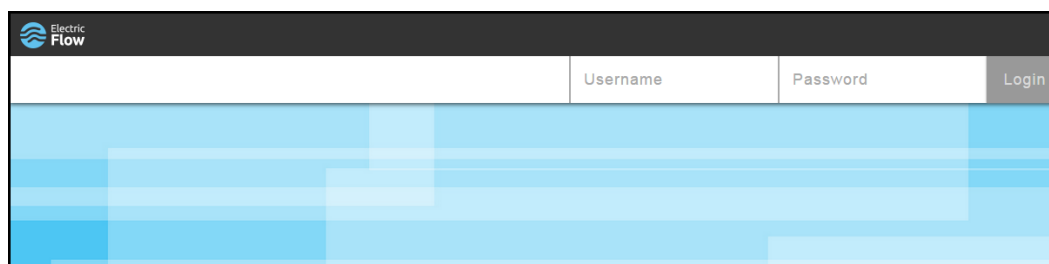
Term	Within ElectricFlow	Outside of ElectricFlow
Application	The application that you design and run (deploy) to produce your software for continuous delivery across different pipelines.	The software, system or application that you build, test, install, implement, release, and deploy using ElectricFlow. This is the end product of using ElectricFlow.
Deploy	Running the application that you designed in ElectricFlow. The end product is your software, system, or application. Deploy is a synonym of run in ElectricFlow.	All the processes or actions to develop and run your software in its environment, including building, testing, implementing, installing, configuring, making changes, and releasing.
Run	Running the application that you designed. The end product is your software, system, or application. <i>Run</i> is a synonym of <i>deploy</i> in ElectricFlow.	All the processes or actions to use software in its environment, including implementing, installing, configuring, debugging, troubleshooting, and releasing.

Logging in to ElectricFlow

1. Enter **http://<ElectricFlow-server>/flow** in a browser window, where <ElectricFlow-server> is the ElectricFlow server IP address or host name.

For example, when you go to <https://123.123.1.222/flow/>, the landing page opens.

Example:

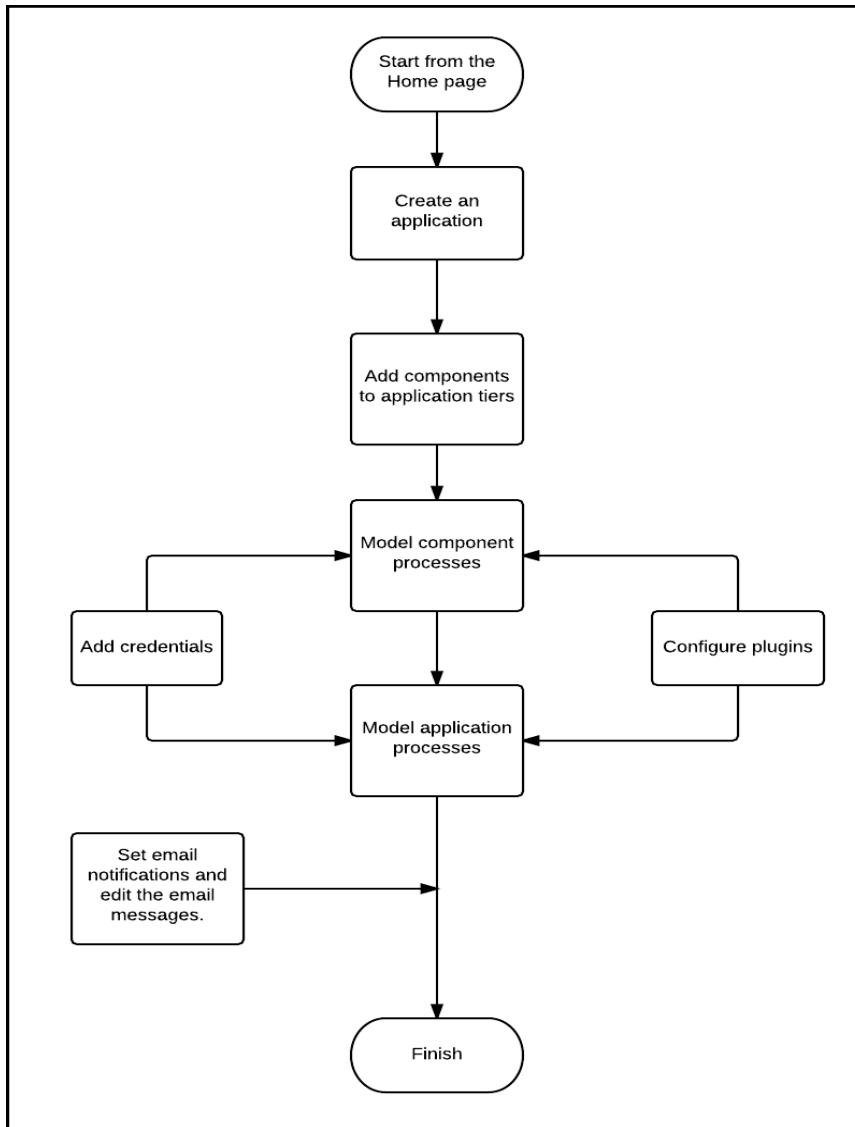


IMPORTANT: For a new installation, the default admin account user name is *admin* and the password is *changeme*. You should change the default admin password as soon as possible.

2. Enter a user name and password.
3. Click **Login**.

The ElectricFlow Home page opens.

Modeling Applications



1. Create an application and application tiers.
2. Add components to the application tiers.
3. Model component processes.
4. Model application processes.
5. (Optional) Set email notifications and edit the email messages.

Creating an Application and Application Tiers

Starting from the Home page:

1. Go to the Applications List by either
 - Clicking the **Applications** launch pad.
 - Clicking the **Menu** button > **Applications**.

The Applications List opens.

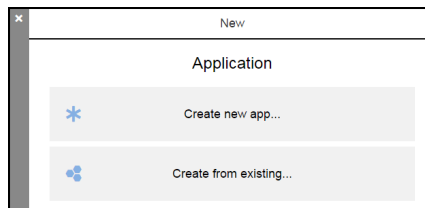
2. Click the **Add +** button in the upper right corner.

Example:



The New Application dialog box appears.

Example:



There are two ways to create an application:

- Click **Create new app** to create a new application.
 - Click **Create from existing** to create an application based on an existing application.
3. Create the application by going to appropriate next step.
 - To create a new application, go to [Creating a New Application](#) on page 197.
 - To create an application based on an existing application, go to the next step.

4. Click **Create from existing** to create an application based on an existing one.

The new application will have the same objects (components, artifacts, and application processes) as the existing application. However, it is not an exact copy of it because you need to configure new tier maps for it.

The **New Application from Existing** dialog box opens.

- a. Select an application.

The **New Application Name** dialog box opens with the name of the application you selected in the **Name** field.

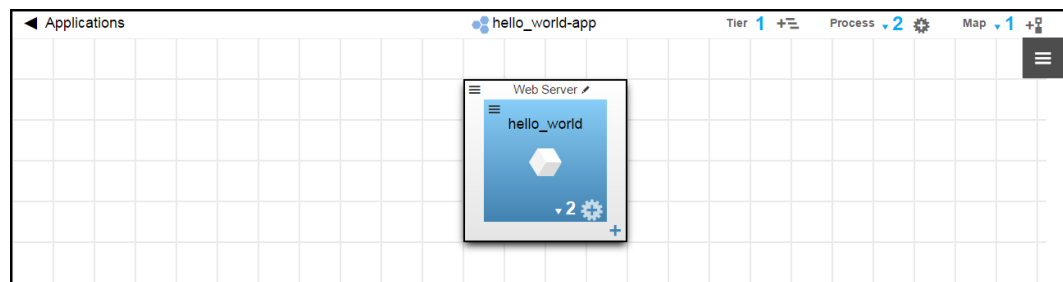
- b. Enter a name in the **Name** field.

It must not match the name of another application in the project.

- c. (Optional) Enter a description of the application in the **Description** field.

- d. Click **OK**.

If you are modeling an application based on an existing application, the Applications Visual Editor displays the same application tiers and components as the existing application with the name that you entered.



Creating a New Application

Starting in the Applications List:

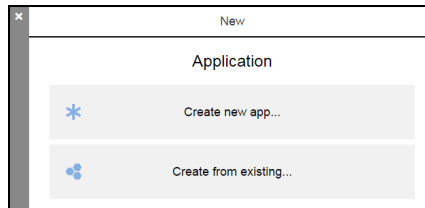
1. Click the **Add +** button in the upper right corner.

Example:



The New Application dialog box appears.

Example:



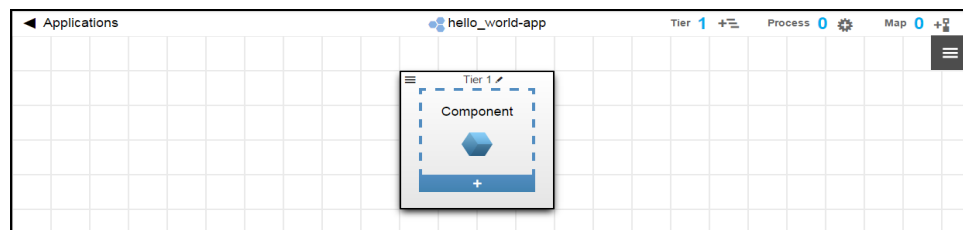
2. Click **Create new app** to create an application.

The **New Application Name** dialog box opens.

- a. Enter a name in the **Name** field.
- b. (Optional) Enter a description of the application in the **Description** field.
- c. Click **OK** to save the settings.

If you are modeling a new application, the Applications Visual Editor displays an application tier called Tier 1 with one component called Component.

Example:



3. Go to [Defining Components](#) on page 360 to set the component details.

Adding Components to the Application Tiers

Starting in the Applications Visual Editor:

1. Click the **Edit** button.

The **Application Tier Details** dialog box opens.

2. Change the name of the tier and click OK.
3. Click the **+** button in the component.

The **New Component** dialog box opens.

4. Enter a name in the **Name** field.
5. (Optional) Enter a description of the component in the **Description** field.
6. Click **Next** to save the settings.

The Component Details dialog box opens.

- Click the **Current Location** field.

A list of available artifact locations appears.

- Select a location and click **Browse**.

The information needed to define the artifact appears below.

- Enter the appropriate information in the fields.

Example:

Component Details

hello_world Description

EC-FileSysRepo Browse

Source: /home/eccloud/sample_dsl/hello_ Required

Artifact: hello_world.html Required

Version: ☒ Latest

☒ Exact 1

Retrieve to Directory:

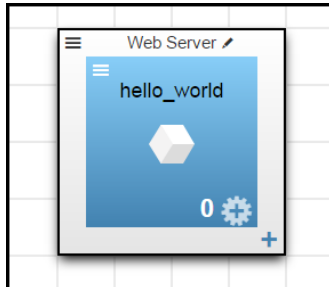
Cancel OK

1	Content Location of the component
2	Component details that vary depending on the Content Location

10. Click **OK**.

The Applications Visual Editor now shows an application tier called Web Server with a component called hello_world.

Example:



11. To add a component to the same tier, click the + button in the lower right corner of the tier.

A new undefined component appears in the tier.

12. Configure this component the same way you did the first one.

Adding a Component

To add component to the same application tier:

1. Click the + button in the lower right corner of the tier.

A new undefined component appears in the tier.

2. Click the + button in the new component to set the component details, following the steps previously described.

Adding an Application Tier

To add an application tier, click the **Add tier** button.

Modeling Component Processes

Starting in the Applications Visual Editor:

1. Click the **Add Process** button in a component to a component process to it.

Example:



The **New Component Process Details** dialog box opens.

- Enter information about the component process in the Component Process Details dialog box.

Example:

The screenshot shows a 'New Component Process Details' dialog box. It features a title bar labeled 'New'. The main content area is titled 'Component Process Details'. Below the title, there are several input fields: a 'Name' text box, a 'Description' text area, a 'Deploy' dropdown menu, a 'Credential' text box containing '0' with an 'Optional' link, a 'Workspace' dropdown menu set to 'default' with an 'Optional' link, and a 'Time limit' section with a text box containing '0', a 'Seconds' dropdown menu, and an 'Optional' link. At the bottom of the dialog are 'Cancel' and 'OK' buttons.

Field	Description and How to Set It
Name	Name of the process step
Description	Description of the process step
Process Type	<p>Type of process. The default is Deploy.</p> <p>To set the process type:</p> <ol style="list-style-type: none"> Click the Type field to select the process type. Select one of these options: <ul style="list-style-type: none"> Deploy—Enables Inventory Tracking. The ElectricFlow server tracks the artifacts deployed to environments. Undeploy—The next time that the process is run, the ElectricFlowserver removes information about the artifacts deployed to environments. Other—Disables Inventory Tracking.

Field	Description and How to Set It
Credential	<p>An object consisting of a user name and password that ElectricFlow uses to run a process step.</p> <p>The dialog box displays the number of credentials for the process step, which are the same credentials that you use with procedures, steps, and schedules in the automation platform.</p> <p>You can only impersonate one credential. See Adding Credentials on page 232 to set the process type.</p>
Workspace	<p>Area in the disk space where the files and results of the job step are stored.</p> <p>To set the workspace, click the Workspace field to open a drop-down list of workspaces in the ElectricFlow platform and select a workspace.</p> <p>For more information about workspaces, go to the "Workspaces and Disk Management" topic. To set the workspace, click Workspace to open a drop-down list of workspaces in the ElectricFlow platform. Select a workspace, and click OK.</p>
Time limit	<p>Maximum length of time that the step is allowed to run. After the time specified, the step is aborted,</p> <p>To set the time limit, enter the time and select the unit of time: seconds, minutes, or hours.</p> <p>For information about time limits for procedure job steps in the ElectricFlow platform, go to the "Job Step Details" topic.</p>

Example:

New

Component Process Details

Deploy

Description

Deploy ▼

Credential

0

Optional >

Workspace

Optional

Time limit

0

Seconds ▼

Optional

Cancel

OK

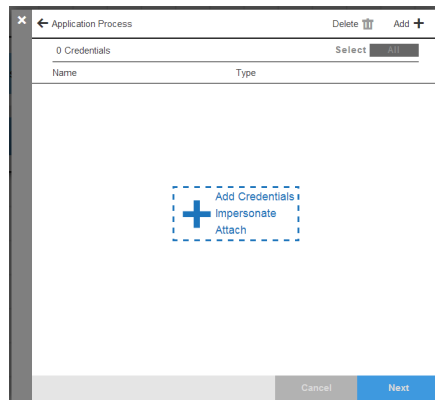
- (Optional) To add credentials, do the following:

IMPORTANT:

When you impersonate a credential, make sure that the impersonated user has the absolute path to the bin directories in the \$PATH environment.

If you define a process step with a command, you must enter the absolute path in the **Post Processor** and **Shell** fields in the Define Step dialog box.

- Click in the **Add Credentials** field.

Example:

- To impersonate one credential, select **Impersonate** in the **Type** field.
- Click the **Select Credential** field to open a drop-down list of credentials for the process step.
- Select a credential.
- Click **OK**.

The **Credentials** dialog box now shows the one credential for impersonation.

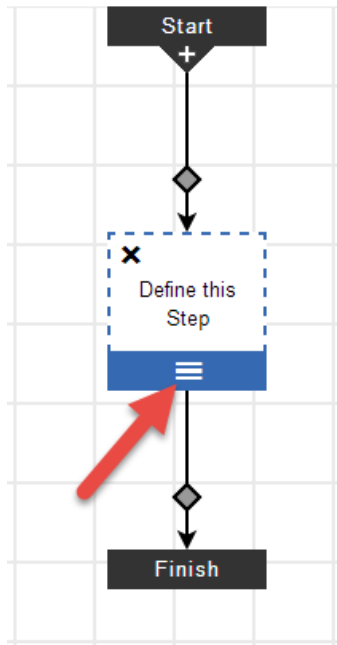
- To attach one or more credential to the process step, select **Attach** in the **Type** field.
- Click the **Select Credential** field to open a drop-down list of credentials for the process step.
- Select a credential.
- Click **OK**.

The **Credentials** dialog box now shows the attached credentials.

- Click **OK**.

The Component Process Visual Editor opens.

5. In the new process step, click the button below "Define this Step" to define it.

Example:

The Component Process Step dialog box opens.

6. Enter information about the step in the dialog box.

Example:

New Step

Component Process Step

Step name: Required

Description:

Credential: >

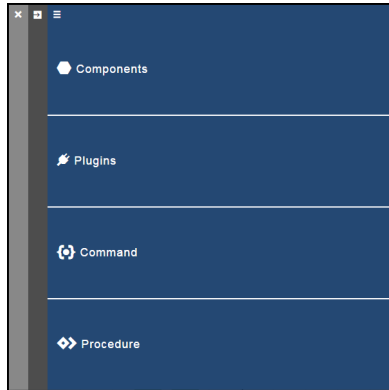
Workspace: ▼

Time limit: ▼

7. Click **Next**.

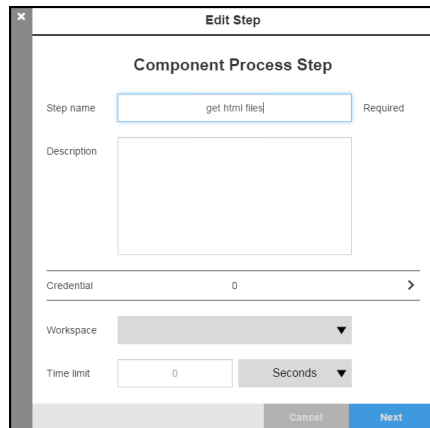
The Process Step dialog box opens.

Example:



8. To define the step, enter information in the dialog boxes that follow.

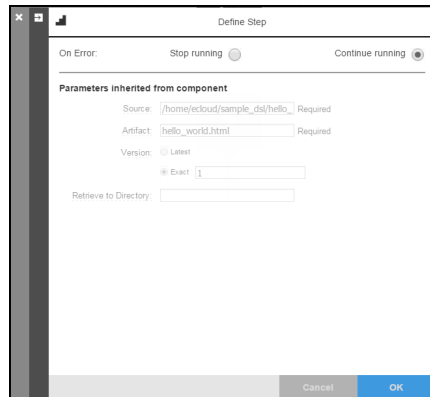
Examples:



The 'Edit Step' dialog box is titled 'Component Process Step'. It contains the following fields:

- Step name:** A text box containing 'get.html.files' with a 'Required' label to its right.
- Description:** A large empty text area.
- Credential:** A dropdown menu showing '0' with a right-pointing arrow.
- Workspace:** A dropdown menu.
- Time limit:** A text box containing '0' and a dropdown menu set to 'Seconds'.

At the bottom are 'Cancel' and 'Next' buttons.



The 'Define Step' dialog box has the following configuration:

- On Error:** Two radio buttons, 'Stop running' (selected) and 'Continue running'.
- Parameters inherited from component:**
 - Source:** '/home/ec2cloud/sample_ds/hello_' with a 'Required' label.
 - Artifact:** 'hello_world.html' with a 'Required' label.
 - Version:** Radio buttons for 'Latest' (selected) and 'Exact' (with a text box containing '1').
 - Retrieve to Directory:** An empty text box.

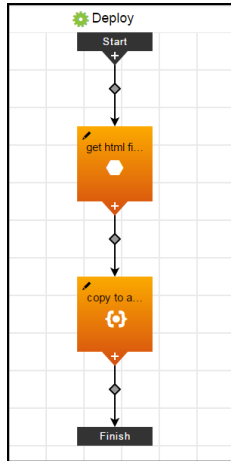
At the bottom are 'Cancel' and 'OK' buttons.

When you are done, the defined step now appears in the process in the Component Process Visual Editor.

9. Define more steps in the process.

You can also drag and drop a step into the process.

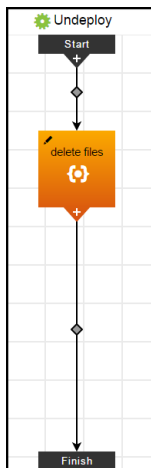
Example:



Modeling More Component Processes

Repeat the steps in the previous section to add additional component processes.

Example:



Modeling Application Processes

Starting in the Applications Visual Editor:

1. Click the **Add process** button in the upper right corner to add an application process.

Example:



The **New Application Process Details** dialog box opens.

2. Enter information in the New Application Process Details dialog box, and click **OK**.

Example:

New

Application Process Details

Name

Description

Credential

0

Optional >

Workspace

default

Optional

Time limit

0

Seconds

Optional

Cancel

OK

Field	Description and How to Set It
Name	Name of the process step
Description	Description of the process step
Credential	<p>An object consisting of a user name and password that ElectricFlow uses to run a process step.</p> <p>The dialog box displays the number of credentials for the process step, which are the same credentials that you use with procedures, steps, and schedules in the automation platform.</p> <p>You can only impersonate one credential.</p>

Field	Description and How to Set It
Workspace	<p>Area in the disk space where the files and results of the job step are stored.</p> <p>To set the workspace, click the Workspace field to open a drop-down list of workspaces in the ElectricFlow platform and select a workspace.</p> <p>For more information about workspaces, go to the "Workspaces and Disk Management" topic.</p> <p>To set the workspace, click Workspace to open a drop-down list of workspaces in ElectricFlow, select a workspace, and click OK.</p>
Time limit	<p>Maximum length of time that the step is allowed to run. After the time specified, the step is aborted,</p> <p>To set the time limit, enter the time and select the unit of time: seconds, minutes, or hours.</p> <p>For information about time limits for procedure job steps in ElectricFlow, go to the "Job Step Details" topic.</p>

- (Optional) To add credentials, do the following:

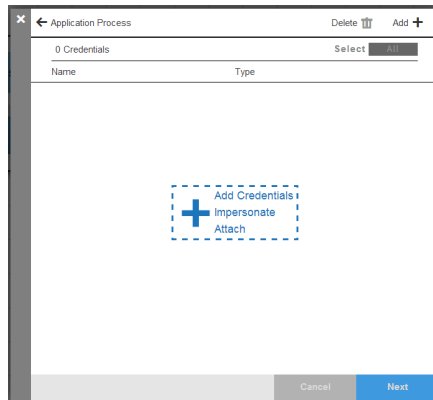
Note:

When you impersonate a credential, make sure that the impersonated user has the absolute path to the bin directories in the \$PATH environment.

If you define a process step with a command, you must enter the absolute path in the **Post Processor** and **Shell** fields in the Define Step dialog box.

- Click in the **Add Credentials** field.

Example:



- To impersonate one credential, select **Impersonate** in the **Type** field.
- Click the **Select Credential** field to open a drop-down list of credentials for the process step.
- Select a credential.
- Click **OK**.

The **Credentials** dialog box now shows the one credential for impersonation.

- To attach one or more credential to the process step, select **Attach** in the **Type** field.
- Click the **Select Credential** field to open a drop-down list of credentials for the process step.
- Select a credential.
- Click **OK**.

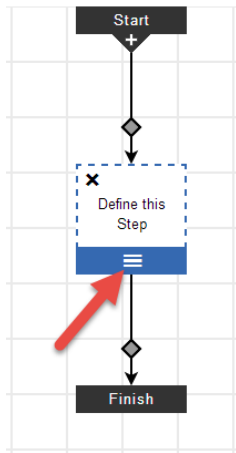
The **Credentials** dialog box now shows the attached credentials.

- Click **OK**.

The Applications Process Visual Editor opens.

5. In the new process step, click the button below "Define this Step" to define it.

Example:



The Application Process Step dialog box opens.

6. Enter information about the step in the dialog box.

Example

New

Application Process Details

Deploy

Description

Credential 0 Optional >

Workspace Optional

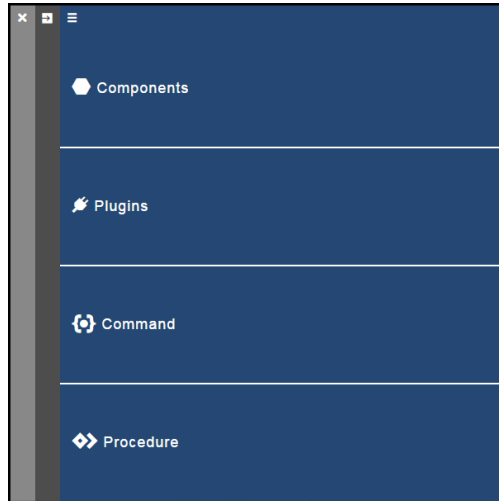
Time limit 0 Seconds Optional

Cancel OK

7. Click **Next**.

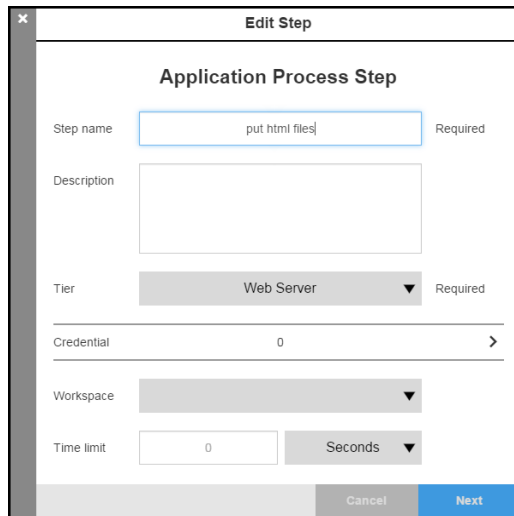
The Process Step dialog box opens.

Example:



8. To define the step, enter information in the dialog boxes that follow.

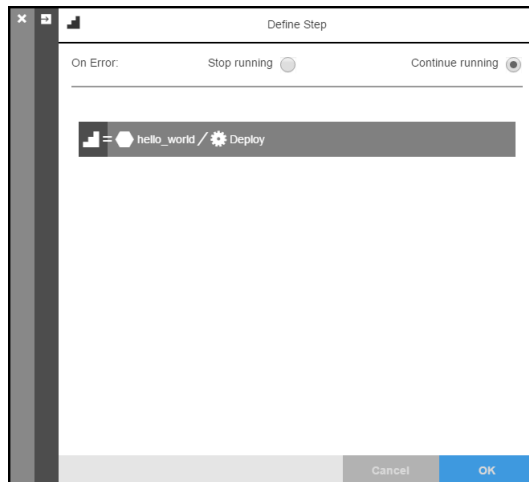
Example:



The 'Edit Step' dialog box is titled 'Application Process Step'. It contains the following fields:

- Step name:** A text box containing 'put.html files'. To its right is the label 'Required'.
- Description:** A large empty text area.
- Tier:** A dropdown menu showing 'Web Server'. To its right is the label 'Required'.
- Credential:** A text box containing '0'. To its right is a right-pointing arrow icon.
- Workspace:** A dropdown menu.
- Time limit:** A text box containing '0' and a dropdown menu showing 'Seconds'.

At the bottom right are 'Cancel' and 'Next' buttons.



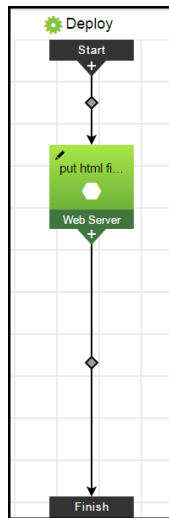
The 'Define Step' dialog box has a title bar with a close button and a maximize button. It contains the following elements:

- On Error:** Two radio buttons: 'Stop running' (unselected) and 'Continue running' (selected).
- Task List:** A single task entry: 'hello_world / Deploy', preceded by a small icon.

At the bottom right are 'Cancel' and 'OK' buttons.

9. When you are done, the defined step now appears in the process in the Applications Process Visual Editor.

Example:



10. Define more steps in the process.

You can also drag and drop a step into the process.

Modeling More Application Processes

Repeat the steps in the previous section to add additional application processes.

Using the Drag and Drop Method to Add Process Steps

How to get to the Application Process Visual Editor:

- In an existing application process:

From the Applications Visual Editor, click the down arrow next to the number-of-application-processes button and select an application.

The Application Process Visual Editor for that application process appears.

- In a new application process:

From the Applications Visual Editor, click the **Add process** button, set the parameters in the **Application Process Details** dialog box, and click **OK**.

The Application Process Visual Editor for the application appears.

How to get to the Component Process Visual Editor:

- In an existing component process:

From the Applications Visual Editor, click the down arrow next to the number-of-component-processes button in a component, and select a component process in the drop-down list.

The Component Process Visual Editor for that component process appears.

- In a new component process:

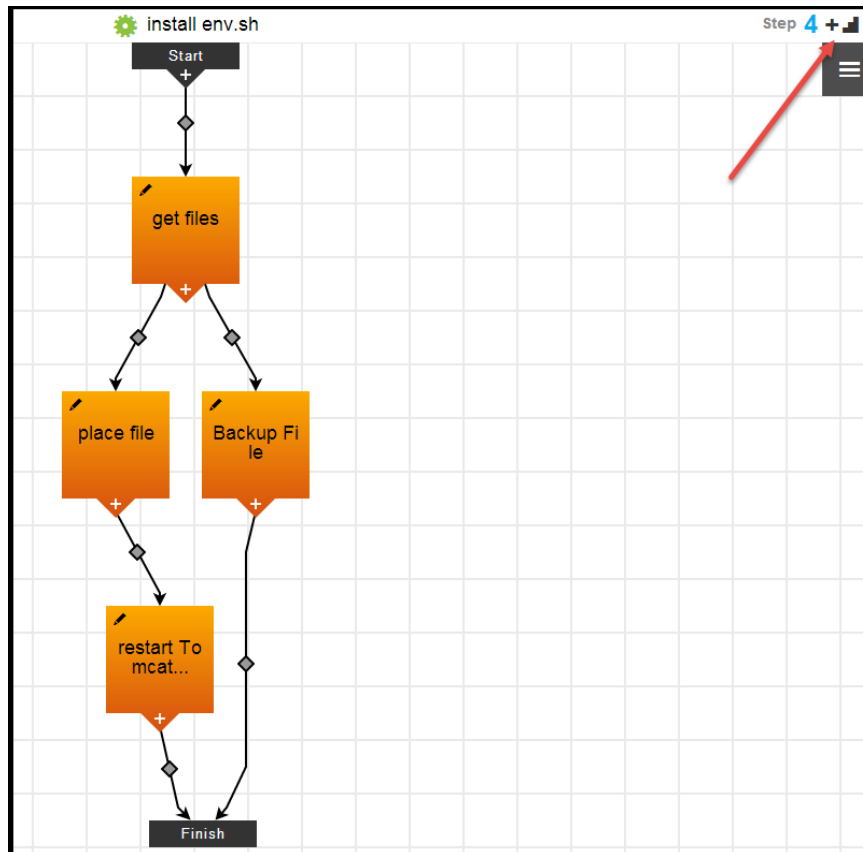
From the Applications Visual Editor, click the **Add process** button in a component, set the parameters in the **Component Process Details** dialog box, and click **OK**.

The Component Process Visual Editor for the component process appears.

To drag and drop a new step in a component or application process:

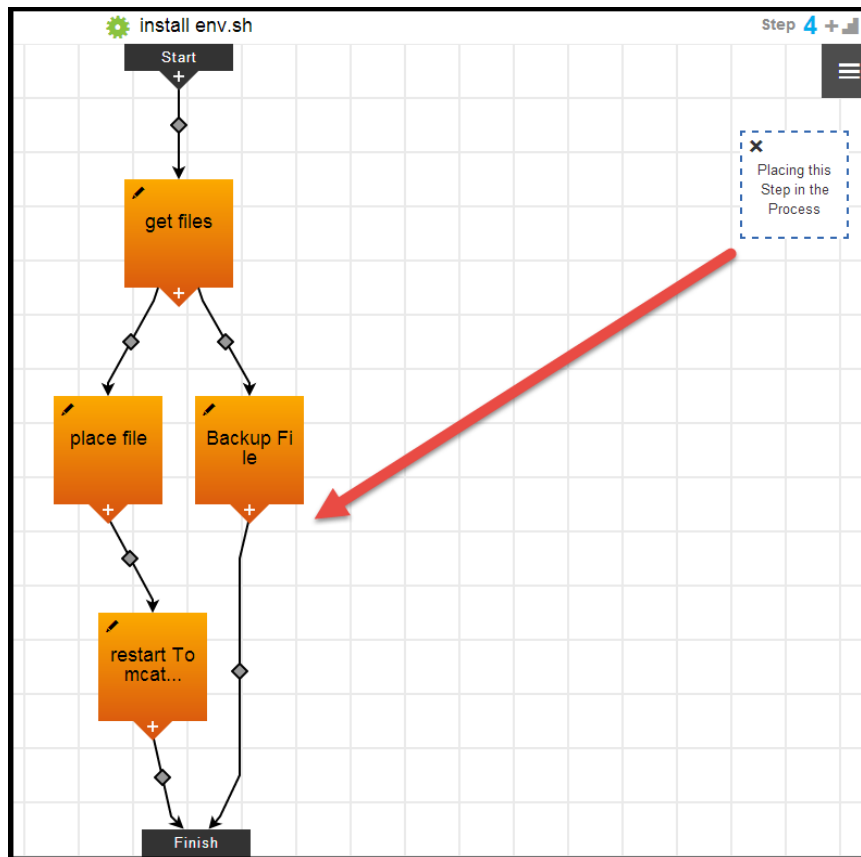
1. Click the **Add step** button in the upper right corner of the Component Process Visual Editor or Application Process Visual Editor.

.A new undefined step appears.



2. Select the new step.

3. Drag the step to where you want to add it in the process.



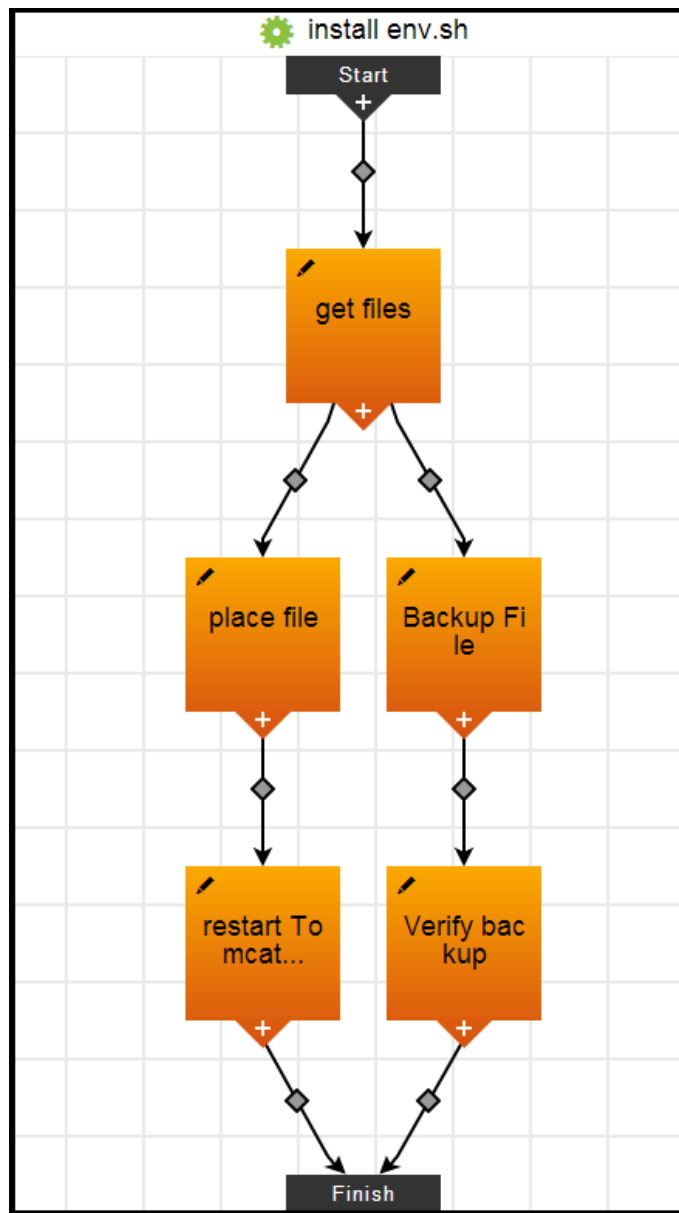
When you are near where you want to add the step in the process, notice that the icon changes shape and the text in it changes to "Dropping this Step in the Process."

4. Drop the step in the process.

The **Component Process Step** dialog box appears.

5. Enter information about the step.

The new step is in the process.

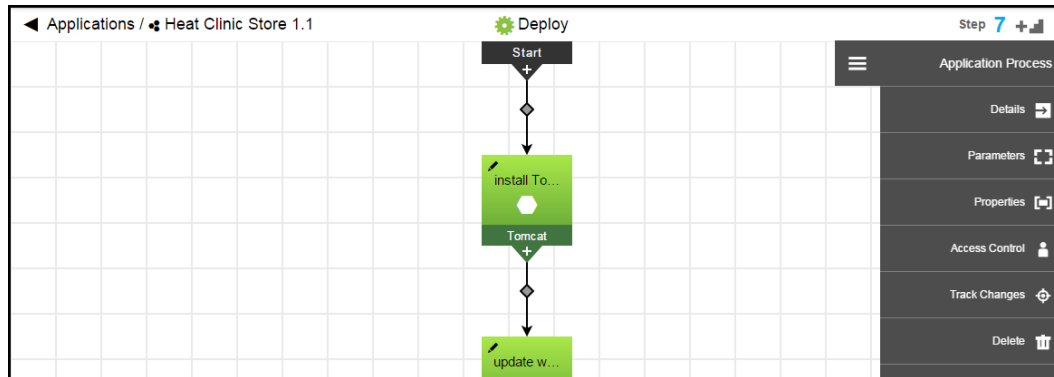


Setting Parameters for Application Processes

Starting in the Application Process Visual Editor:

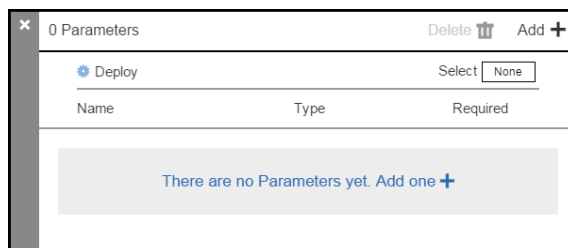
1. Click the **Menu** button.
2. Click **Parameters**.

Example:



The **Parameters** dialog box opens.

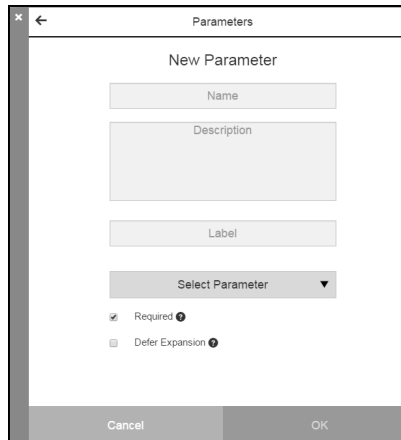
Example:



3. Click **There are no Parameters yet. Add one +**.

The **New Parameter** dialog box opens.

Example:



The screenshot shows a 'Parameters' window with a 'New Parameter' dialog box open. The dialog box has a title bar with a close button and a back arrow. Inside, the title 'New Parameter' is centered. Below the title are four input fields: 'Name', 'Description', 'Label', and 'Select Parameter' (a dropdown menu). At the bottom of the dialog are two checkboxes: 'Required' (checked) and 'Defer Expansion' (unchecked). The dialog box has 'Cancel' and 'OK' buttons at the bottom.

4. Enter the following information:

- **Parameter Name**—Name of the parameter.
- **Description**—This is optional.
- **Parameter Label**—This is optional.
- **Select Parameter**—Parameter type
- If the parameter is required, select the **Required** check box.
- If the parameter value contains \$ [] and you want ElectricFlow to interpret it literally (not as a parameter reference), select the **Defer Expansion** check box.

Depending on the parameter type that you select, other fields appear in the **New Parameters** dialog box.

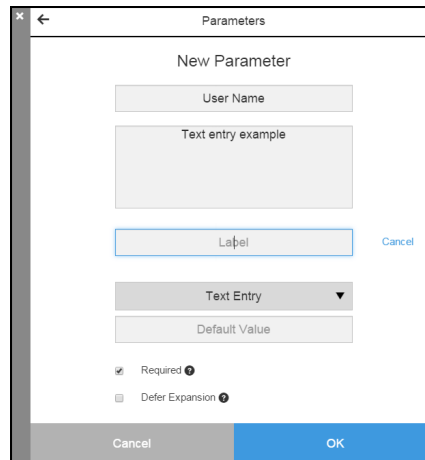
5. Enter information in the fields in Step 4.

If you select **Text Entry** as the parameter type, the **Default Value** field appears.

- a. (Optional) Enter a value in the **Default Value** field.

You do not have to enter a value in this field.

Example:



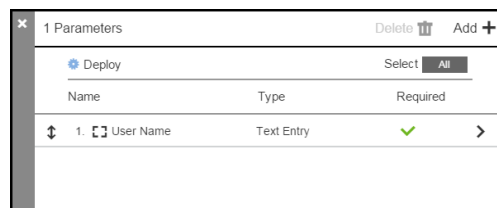
The screenshot shows a 'Parameters' dialog box with a 'New Parameter' section. It contains the following fields and options:

- User Name**: A text input field.
- Text entry example**: A larger text input field.
- Label**: A text input field.
- Text Entry**: A dropdown menu.
- Default Value**: A text input field.
- Required**: A checked checkbox.
- Defer Expansion**: An unchecked checkbox.
- Buttons**: 'Cancel' and 'OK' buttons at the bottom.

- b. Click **OK**.

The **Parameters** dialog box opens and shows the new parameter in the list.

Example:



The screenshot shows the 'Parameters' dialog box with a list of parameters. The list has columns for Name, Type, and Required. The first parameter is 'User Name' of type 'Text Entry' and is required.

Name	Type	Required
1. User Name	Text Entry	✓

6. Click **Add +** to add a parameter.

7. Enter information in the fields in Step 4.

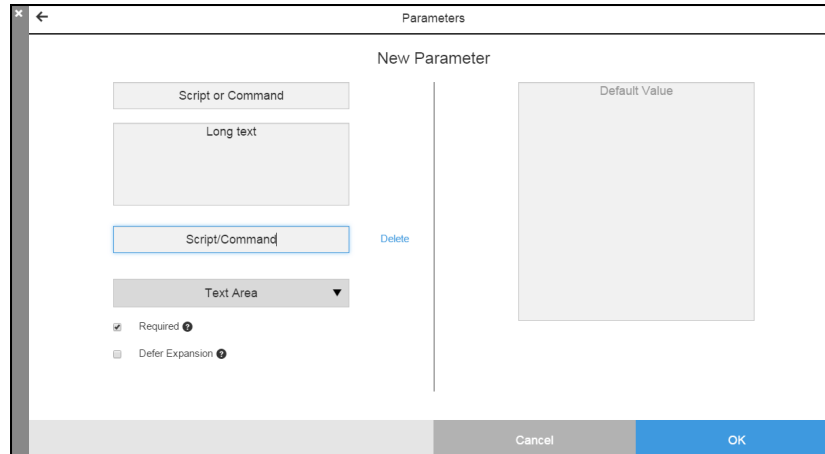
If you select **Text Area** as the parameter type, the **Default Value** field appears.

- a. (Optional) Enter a value in the **Default Value** field.

You do not have to enter a value in this field.

- b. Enter the Click **OK**.

Example:



The screenshot shows the 'Parameters' dialog box with a 'New Parameter' section. On the left, there are input fields for 'Script or Command', 'Long text', and 'Script/Command', followed by a 'Text Area' dropdown menu. Below these are checkboxes for 'Required' (checked) and 'Defer Expansion' (unchecked). On the right, there is a large 'Default Value' text area. A 'Delete' link is positioned between the input fields and the 'Default Value' area. At the bottom, there are 'Cancel' and 'OK' buttons.

The **Parameters** dialog box opens and shows the new parameter in the list.

8. Enter information in the fields in Step 4. If you select **Dropdown Menu** as the parameter type, the **Default Value** field and ways to add the menu options appear.

- a. (Optional) Enter a value in the **Default Value** field.

You do not have to enter a value in this field.

- b. On the right side of the dialog box, enter the menu options.

- c. Click **OK**.

The **Parameters** dialog box opens and shows the new parameter in the list.

Example:

The screenshot shows the 'Parameters' dialog box with the 'New Parameter' form. The form is divided into two main sections. The left section contains fields for 'Stage', 'Stage of the deployment process', 'Label', 'Dropdown Menu' (set to 'Dropdown Menu'), and 'Default Value'. Below these fields are checkboxes for 'Required' (checked) and 'Defer Expansion' (unchecked). The right section shows three radio buttons for 'Enter options' (selected), 'Load options from list', and 'Load options from property sheet'. Below these is an 'Add option +' button. A list of three options is shown: 1. 'Development' / 'Dev', 2. 'Quality' / 'QA', and 3. 'Release' / 'Prod'. The 'Prod' option is highlighted. At the bottom are 'Cancel' and 'OK' buttons.

The **Parameters** dialog box opens and shows the new parameter in the list.

9. Enter information in the fields in Step 4.

If you select **Radio Selector** as the parameter type, the **Default Value** field and ways to add the menu options appear.

- a. (Optional) Enter a value in the **Default Value** field.
You do not have to enter a value in this field.
- b. On the right side of the dialog box, enter the menu options.
- c. Click **OK**.

Example:

The screenshot shows the 'Parameters' dialog box with the 'New Parameter' tab active. On the left, a list of parameters includes 'Ranking', 'Ranking in the queue', 'Priority', 'Radio Selector' (selected), and 'Normal'. The 'Radio Selector' parameter is configured with the 'Required' checkbox checked and 'Defer Expansion' unchecked. On the right, the 'Enter options' radio button is selected. Below it, three menu options are listed: 1. 'High', 2. 'Normal', and 3. 'Low'. The 'Low' option is currently selected and highlighted with a blue border. At the bottom of the dialog are 'Cancel' and 'OK' buttons.

The **Parameters** dialog box opens and shows the new parameter in the list.

10. Enter information in the fields in Step 4.

If you select **Checkbox** as the parameter type, the **Default Value** field and values for the check box appear.

- a. (Optional) Enter a value in the **Default Value** field.
You do not have to enter a value in this field.
- b. On the right side of the dialog box, enter the values.
- c. Click **OK**.

Example:

The screenshot shows the 'Parameters' dialog box with the 'New Parameter' tab active. On the left, a list of parameters includes 'Ranking', 'Ranking in the queue', 'Priority', 'Radio Selector', and 'Normal'. The 'Radio Selector' parameter is selected, and its details are shown on the right. The 'Enter options' radio button is selected. Below it, three options are listed: '1. High', '2. Normal', and '3. Low'. The 'Low' option is currently selected. At the bottom, there are 'Cancel' and 'OK' buttons. The 'Required' checkbox is checked, and the 'Defer Expansion' checkbox is unchecked.

The **Parameters** dialog box opens and shows the new parameter in the list.

11. Enter information in the fields in Step 4. If you select **Credentials** as the parameter type, the **Default Value** field appears.
 - a. (Optional) Enter a value in the **Default Value** field.
You do not have to enter a value in this field.
 - b. Click **OK**.

Example:

The **Parameters** dialog box opens and shows the new parameter in the list.

Example:

Name	Type	Required	
1. [icon] User Name	Text Entry	✓	>
2. [icon] Script/Command	Text Area		>
3. [icon] Stage	Dropdown menu	✓	>
4. [icon] Priority	Radio Selector	✓	>
5. [icon] QA Verification Required	Checkbox		>
6. [icon] User Name and Password	Credential	✓	>

Setting and Modifying the Parameter Label

Starting in the **New Parameter** dialog box:

Example:

Parameters

New Parameter

QA Verification Required

Description

Label

Checkbox

Default Value

☐ Required

☐ Defer Expansion

Value when unchecked

No

Value when checked

Yes

☐ Initially checked

Cancel OK

1. Add a label in the **Label** .

Example:

New Parameter

QA Verification Required

Description

QA Verify Delete

Checkbox

Default Value

☐ Required

☐ Defer Expansion

Value when unchecked

No

Value when checked

Yes

☐ Initially checked

This is an optional task when you set parameters.

If you enter a label, it appears in the UI form when you deploy the application. If you do not enter a label, the parameter name appears in the UI form when you deploy the application.

2. Click **OK**.

3. To modify the label:
 - a. Open the **New Parameter** dialog box.
 - b. Clear the **Label** field.
 - c. Enter a new label.

Example:

The screenshot shows the 'New Parameter' dialog box. The left pane has the following elements: a text field 'QA Verification Required', a large text area 'Description', a text field 'Verify by QA' (highlighted with a red box), a dropdown menu 'Checkbox', and a text field 'Default Value'. The right pane has: a text field 'Value when unchecked' with 'No' entered, a text field 'Value when checked' with 'Yes' entered, and a checkbox 'Initially checked'. At the bottom left are two checkboxes: 'Required' and 'Defer Expansion', both with question mark icons.

- d. Click **OK** to save the change.

4. To delete the label:
 - a. Click **Delete** next to the **Label** field.

Example:

The screenshot shows a 'Parameters' window with a 'New Parameter' section. On the left, there are input fields for 'QA Verification Required', 'Description', 'Verify by QA', 'Checkbox', and 'Default Value'. Below these are checkboxes for 'Required' and 'Defer Expansion'. On the right, there are input fields for 'Value when unchecked' (containing 'No') and 'Value when checked' (containing 'Yes'), along with an 'Initially checked' checkbox. A blue 'Delete' button is highlighted with a red rectangle next to the 'Verify by QA' field. At the bottom are 'Cancel' and 'OK' buttons.

The text in the Label field changes.

Example:

This screenshot shows the same 'New Parameter' dialog box, but with a confirmation prompt. A blue button labeled 'Yes, delete this Label' is highlighted with a red rectangle, next to a smaller blue 'Cancel' button. The rest of the dialog box, including the input fields and the bottom 'Cancel'/'OK' buttons, remains the same as in the previous screenshot.

- b. Click in the **Label** field.

The label disappears.

Example:

Parameters

New Parameter

QA Verification Required

Description

Label

Checkbox

Default Value

Value when unchecked

No

Value when checked

Yes

☐ Initially checked

Required ?

Defer Expansion ?

Cancel OK

- c. Click **OK**.

The **Parameters** dialog box opens. The parameter name now appears in the name column.

Example:

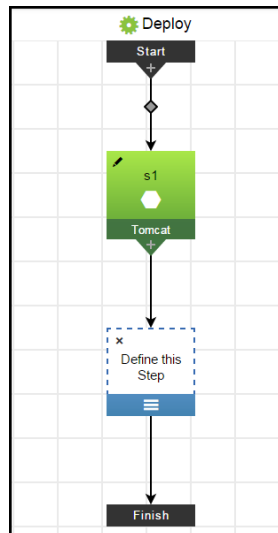
6 Parameters			Delete	Add
Deploy		Select	All	
Name	Type	Required		
1. User Name	Text Entry		>	
2. Script/Command	Text Area		>	
3. Stage	Dropdown menu		>	
4. Priority	Radio Selector		>	
5. QA Verification Required	Checkbox		>	
6. User Name and Password	Credential		>	

Looking Up Parameters

To apply parameters to an application or component process step, starting in the Application Process Visual Editor:

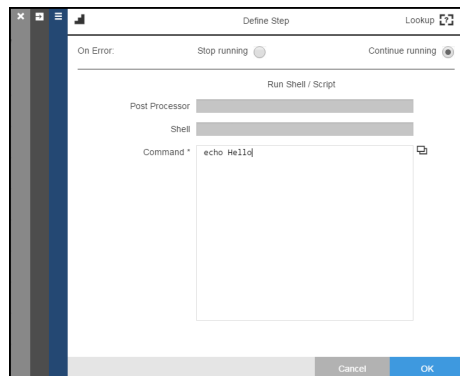
1. Add a new step to the process.

Example:



2. Define the process step in the **Define Step** dialog box.

Example:

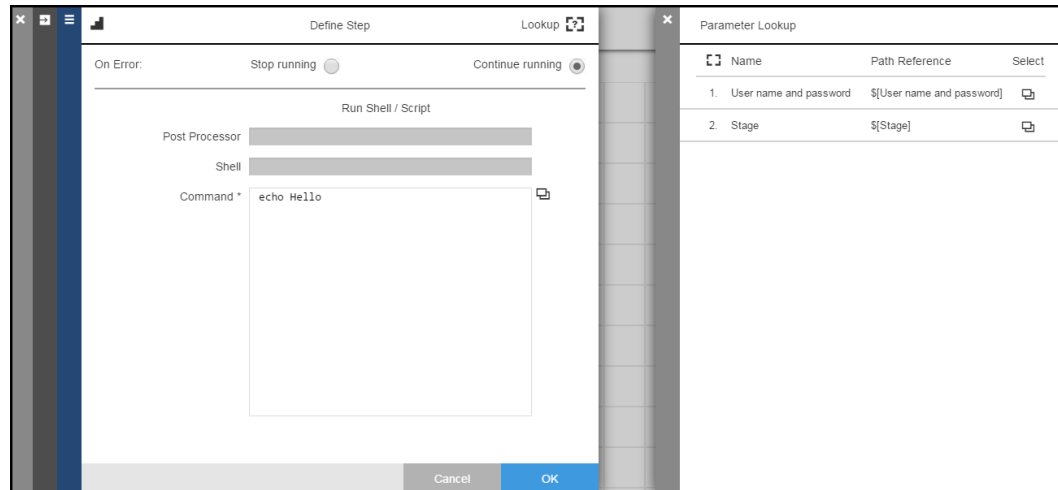


- When you define the process step with a plugin (**Plugins**), command or script (**Command**), or project (**Procedure**), click the **Lookup** button to open the **Parameter Lookup** dialog box.

Example:



The **Parameter Lookup** dialog box opens.



- Choose a parameter and click the **Copy** button to copy the path reference.

A message appears in the row : *<Parameter Name>* has been copied.

The **Define Step** dialog box now has a **Parameter** field.

- Click in the **Parameter** field and paste the path reference that you copied in it.
- Repeat the previous two steps to apply another parameter to the process step.
- Click **OK**.

Adding Credentials

How to get here: From the **Component Process Step** or the **Application Process Step** dialog box, click **>**.The **Credentials** dialog box opens.

You can attach one or more credentials to component process steps and application process steps.

You attach only one credential for impersonation on the following:

- Component process
- Component process step

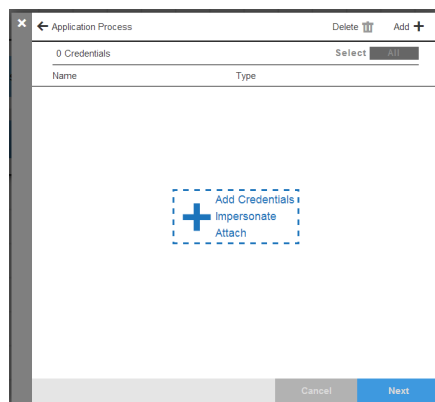
- Application process
- Application process step

IMPORTANT:

When you impersonate a credential, make sure that the impersonated user has the absolute path to the bin directories in the \$PATH environment.

If you define a process step with a command, you must enter the absolute path in the **Post Processor** and **Shell** fields in the Define Step dialog box.

1. Click in the **Add Credentials** field.

Example:

2. To impersonate one credential, select **Impersonate** in the **Type** field.
3. Click the **Select Credential** field to open a drop-down list of credentials for the process step.
4. Select a credential.
5. Click **OK**.

The **Credentials** dialog box now shows the one credential for impersonation.

6. To attach one or more credential to the process step, select **Attach** in the **Type** field.
7. Click the **Select Credential** field to open a drop-down list of credentials for the process step.
8. Select a credential.
9. Click **OK**.

The **Credentials** dialog box now shows the attached credentials.

Using Plugins

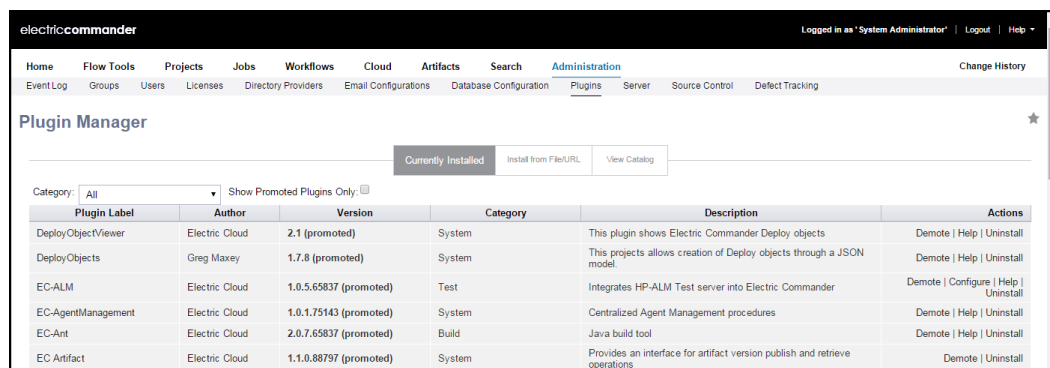
When you want to use a plugin to define your application or component process step, all of the supported plugins appear in the Plugin Manager in the **Currently Installed** tab in the automation platform. However, you may want to see only the list of plugins that apply to your group or organization, such as Apache Subversion

(SVN) and Git plugins. You can remove any plugins that you are not using from the **Currently Installed** tab. You can re-install them later if you need them.

To remove plugins from the **Currently Installed** tab:

1. In the Home page, click the **Main menu** button and then click **Admin > Plugins**.

The Plugin Manager opens.



2. In the **Currently Installed** tab, select a plugin in the list.
3. In the Actions column for the selected plugin, click **Demote**.

The page refreshes.

The plugin is now inactive but is still in the list. If you want to use this plugin, click **Promote** to make it active.

4. If you want to remove the plugin from this list, click **Uninstall** to remove it from your system.

For more details, go to the automation platform Help > **Web Interface Help > Plugin Manager**. This page describes what you can do in the Plugin Manager, including how to install a new version of a plugin or add a new plugin.

You can see all of the plugins available from Electric Cloud in the **View Catalog** tab on the **Administration > Plugins** page. To show a list of plugins that you can install directly from Electric Cloud, do the following:

1. Click **Install** in the **Action** column.
2. Go to the **Currently Installed** tab.
3. Choose a plugin and click **Promote**.

The new plugin is now available for use in your system.

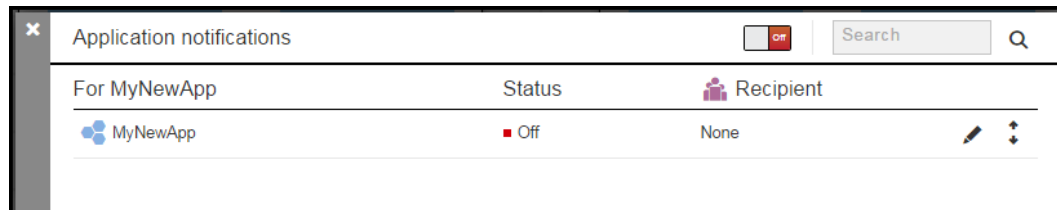
Setting Email Notifications

Review these guidelines before setting notifications:

- Configure notifications for application processes and application process steps (type: process).

When you open the Applications notifications dialog box and expand the application process details, only the application process steps (type: process) appear in the list.

- New email notifications are disabled in the application, its application processes, and the application process steps (type: process) before you configure them.



- You configure notifications in the "Application notifications / edit" dialog box.

IMPORTANT: The first time that you set notifications in this dialog box, the Notifications toggle changes to **On**. After you enter notification settings and click **OK**, email notifications are enabled at that level.

Go to [Application Notifications Dialog Box](#) on page 392 for information about how to use the "Application notifications" and "Application notifications /edit" dialog boxes.

By default, the application expects that the user creates an email configuration called "default." The email configuration defaults to the server property `/server/ec_deploy/ec_defaultEmailConfiguration`, which is set to "default."

If you want to use a different name for the email configuration, change the value of `/server/ec_deploy/ec_defaultEmailConfiguration` to the new email configuration name.

Starting from the Home page, to set email notifications:

1. Go to the Applications List.

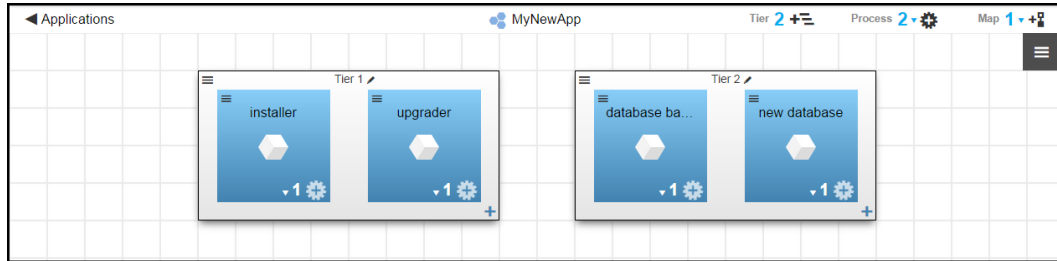
Example:

4 Applications					Select	Delete	Add
1	Arthur 23	1 Component	1 Application Process	1 Tier Map			
2	Heat Clinic Store 1.1	5 Component	2 Application Process	3 Tier Map			
3	MyNewApp	4 Component	2 Application Process	1 Tier Map			
4	Test	1 Component	1 Application Process	1 Tier Map			

2. Select an application.

The Applications Visual Editor opens.

Example:



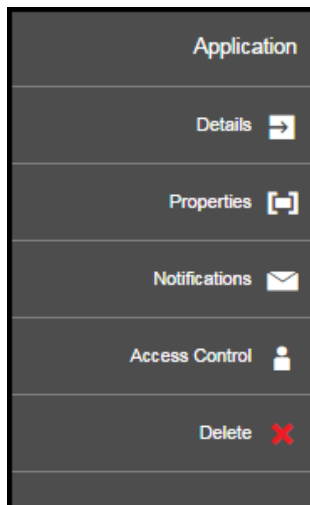
3. Click the **Menu** button.

Example:



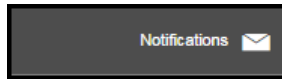
The Applications menu opens.

Example:



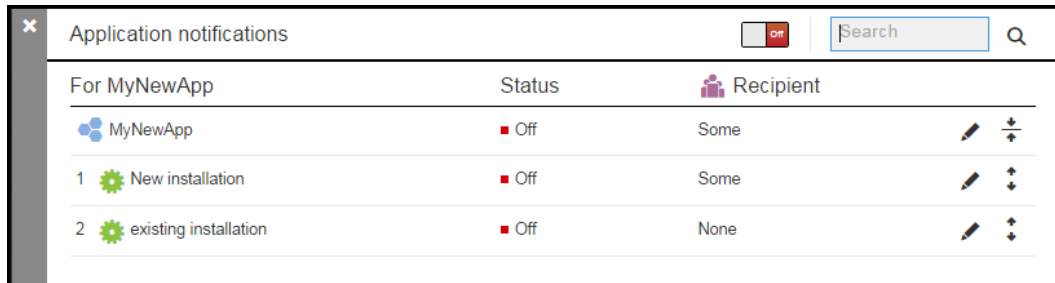
- Click **Notifications** to add a new application.

Example:



The Application notification dialog box opens.

Example:



- Configure email notifications for the application, an application process, or a process step.

You can configure one or more notifications in an application process or other object.

Configuring recipients

In the **Who** field, you add users or groups who are configured and managed in the ElectricFlow platform or email addresses.

When you start typing a user name, group name, or email addresses, a list of names or email addresses appear that match what you are typing.

Example:

Application notifications / edit

backup On

Who	When	Where
Add users, groups, or email addresses:	Event:	Environment:
a	Both Failed and Success...	All
admin (?) admin-asia (?) admin-aus (?) admin-uk (?) admin-us (?) jadams (?) sclaus (?)		

If one of the suggestions matches the name or email address, select it, or continue typing. You can add more than one name or email address.

Example:

Application notifications / edit

backup On

Who	When	Where
Add users, groups, or email addresses:	Event:	Environment:
admin admin-asia sclaus jadams userX@gmail.com	Both Failed and Success...	All
DevT200@gmail.com		

Configuring the event that triggers the notification

In the **When** field, you select the event that triggers a notification to be sent to the recipients in the **Who** field. The default is **Both Failed and Successful**. Click in the **When** field to select the event for the notification.

Example:

Application notifications / edit

backup On

Who	When	Where
Add users, groups, or email addresses:	Event:	Environment:
admin admin-asia sclaus jadams userX@gmail.com	Both Failed and Successful ▼	All
DevT200@gmail.com	Run Failed	
	Run Successful	
	Both Failed and Successful	

Configuring the environments where the notification applies

In the **Where** field, you select the environments to which the notifications apply. Click in the **Where** field to select the environments, which are the environments to which the application is mapped in the tier map.

Example:

Application notifications / edit

backup On

Who	When	Where
Add users, groups, or email addresses:	Event:	Environment:
admin admin-asia sclaus jadams userX@gmail.com	Run Successful ▼	All
DevT200@gmail.com		hc-store dev

6. Select and edit the email notification message.

7. Add another notification for the the application, an application process, or a process step.

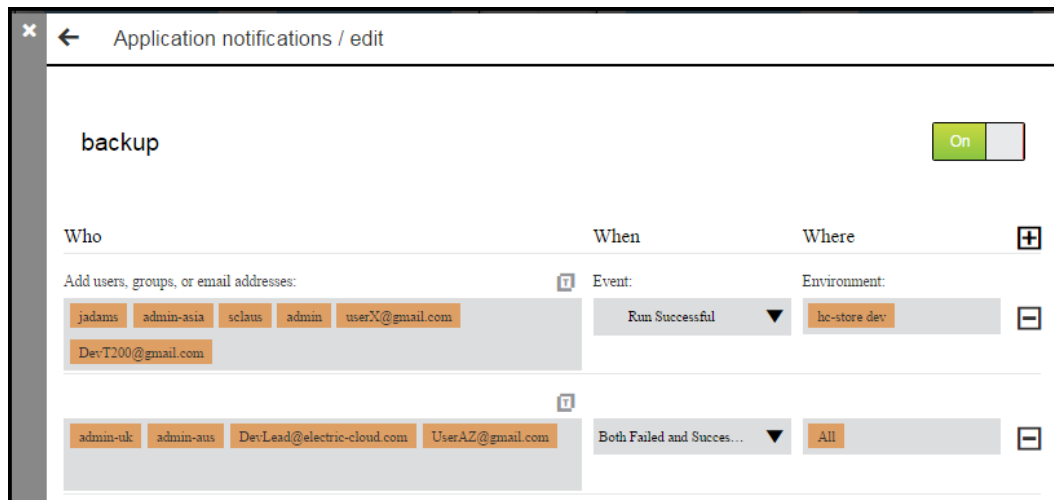
Click the **Add Notifications** button to add a new notification.

Example:



After you have added your email notifications, click **OK** to save the settings and return to the Application notifications dialog box.

Example:



The screenshot shows a dialog box titled "Application notifications / edit". Inside, the notification name "backup" is at the top right, next to an "On" toggle switch. Below this is a table with three columns: "Who", "When", and "Where". The "Who" column has a sub-header "Add users, groups, or email addresses:" and contains two rows of email addresses: "jadamis admin-asia sclaus admin userX@gmail.com" and "DevT200@gmail.com". The "When" column has a sub-header "Event:" and contains two rows: "Run Successful" and "Both Failed and Succes...". The "Where" column has a sub-header "Environment:" and contains two rows: "bc-store dev" and "All". There are plus and minus icons for adding and removing rows in each column.

Who	When	Where
Add users, groups, or email addresses:		
jadamis admin-asia sclaus admin userX@gmail.com	Run Successful	bc-store dev
DevT200@gmail.com		
admin-uk admin-aus DevLead@electric-cloud.com UserAZ@gmail.com	Both Failed and Succes...	All

8. (Optional) Enable email notifications for the application, application processes, or process steps that are not already enabled.

To enable email notifications at the application level:

- Click the Notifications toggle and change it to **On**.

The status of the application changes to **On**.

Example:

For MyNewApp	Status	Recipient
MyNewApp	On	None
1 New installation	Off	None
2 existing installation	Off	None

- Click the **Edit** button to open the **Application notifications / edit** dialog box.

The **Application notifications / edit** dialog box appears. The Notification toggle changes to **On**.

Example:

MyNewApp On

Who: Add users, groups, or email addresses:

When: Event: Both Failed and Success... ▼

Where: Environment: All [-]

To enable notifications at the application process and process step levels, go to the **Application Notifications / edit** dialog box for the specific process or process step.

The dialog box opens, and the Notifications toggle is now **On**.

Example:

When you enter notification settings in the dialog box and click **OK**, the settings are saved. The **Application notifications** dialog box appears and now shows that the application process status is **On**.

Example:

For MyNewApp	Status	Recipient
MyNewApp	On	Some
1 New installation	On	Some
2 existing installation	Off	None

Selecting and Editing Email Messages

Starting in the "Application notifications / edit" dialog box:

1. Click the **Template** button .

A drop-down box opens.

Example:

Application notifications / edit

backup On

Who	When	Where
Add users, groups, or email addresses:	Event:	Environment:
jadams admin-asia sclaus admin userX@gmail.com	Run Successful	hc-store dev
DevT200@gmail.com		
admin-uk admin-aus	Both Failed and Succes...	All

Apply email message template

Default Success notific...

Cancel OK

2. Click the down arrow to open the list of email message templates that can apply to the application.

3. Select a template.

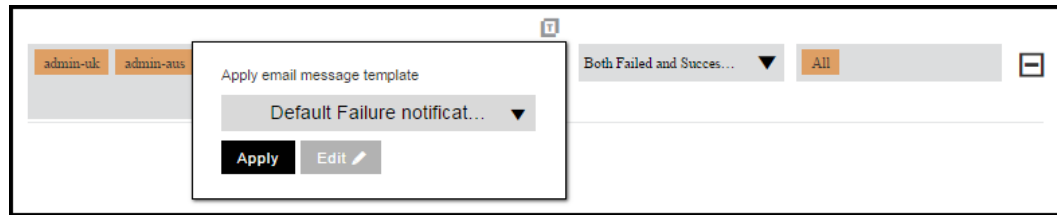
Example:

If the template is the current template applied to notification, the name of the template appears in dialog box.

Example:

If the template is not the current template, the **Apply** and **Edit** buttons appear in the dialog box.

Example:



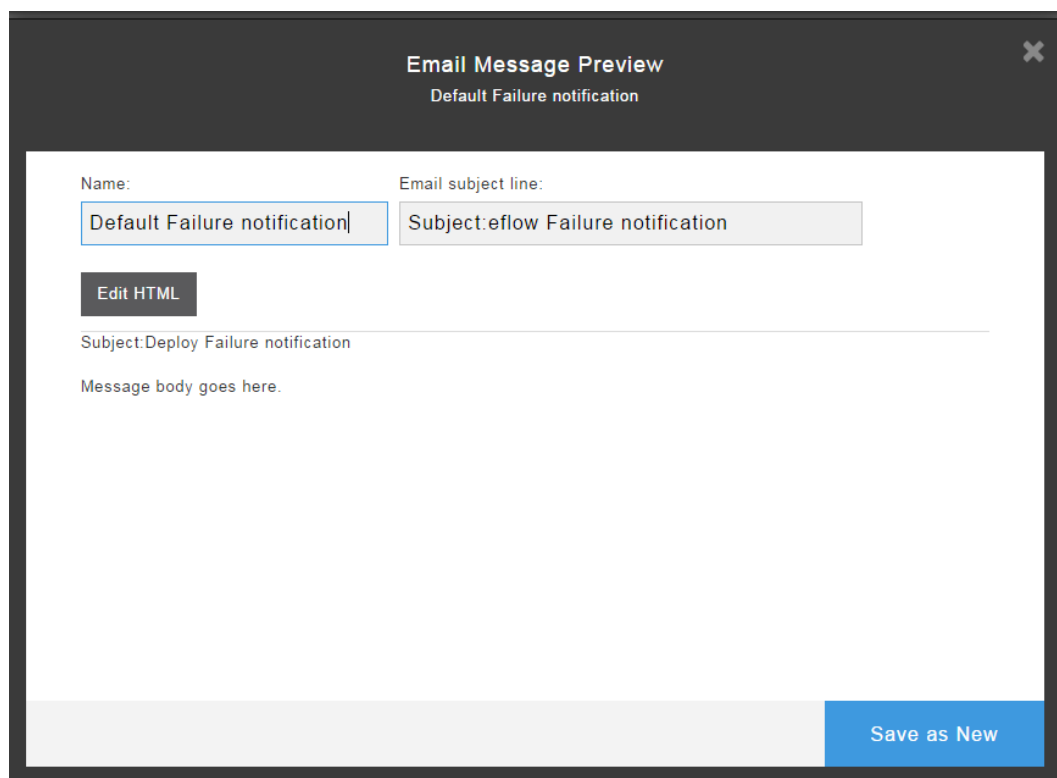
4. If you want to use the template that you selected instead of the current one and do not want to change it, skip the remaining steps in this task.
5. If you want to apply a different template or edit the template that you selected, do the remaining steps in this task.

If you click **Apply** to use the template as is, skip the remaining steps.

If you click **Edit** to modify the template to fit your needs, go to the next step.

The Email Message Preview dialog box appears.

Example:



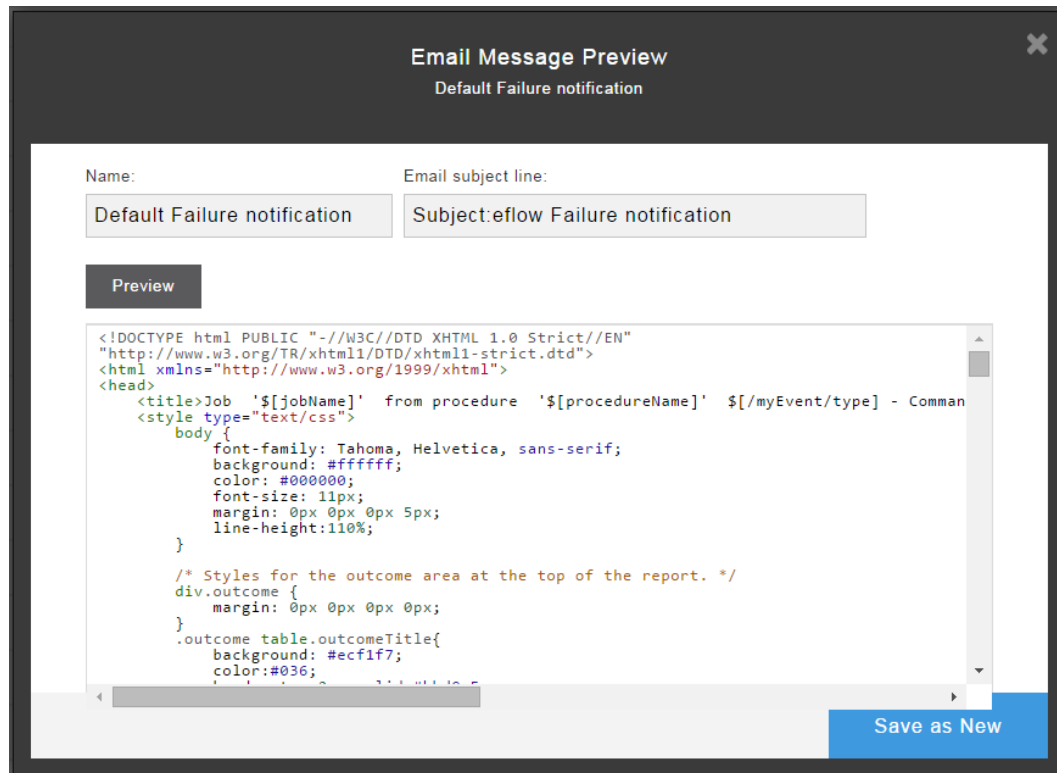
6. To edit the template:

Change the name of the template in the **Name** field.

Change the subject of the email in the **Email subject line**.

To modify the body of the email message, click **Edit HTML** and edit the HTML code.

Example:



7. Click **Preview**.

8. To save your changes:

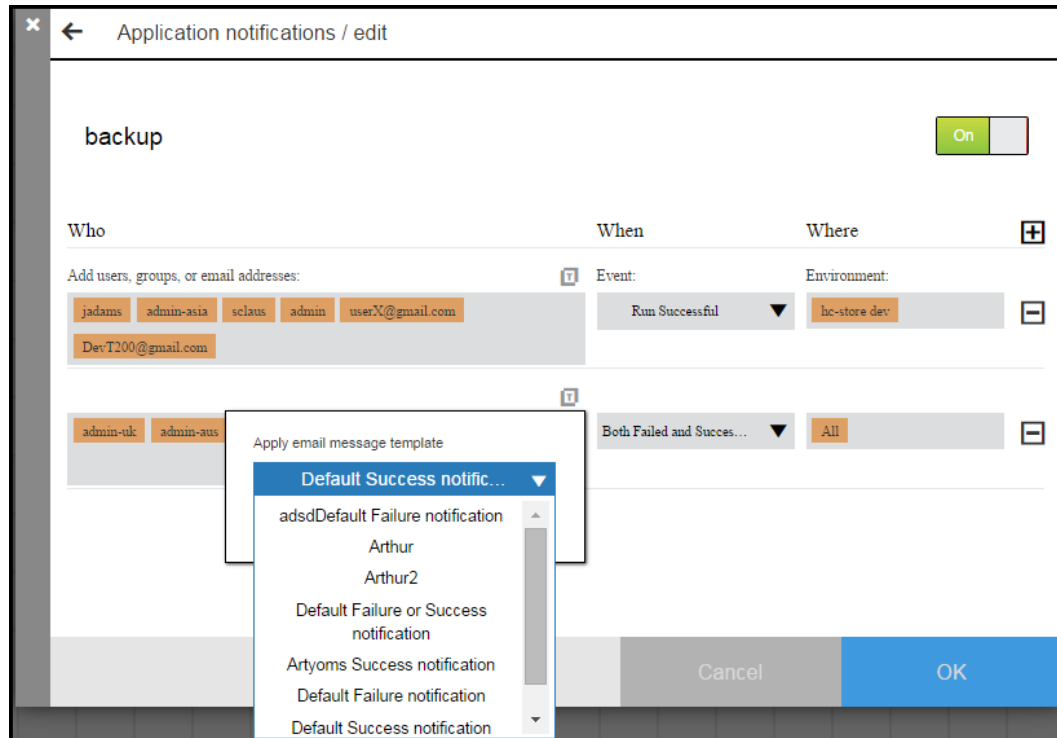
- Click **Save Changes** to save the change in an existing template
- Click **Save as New** to save the template as a new template.

The "Application notifications / edit" dialog box re-appears.

- Click the down arrow to open the list of email message templates that can apply to the application.

The new email message template is in the list.

Example:



- Click **OK** to save the settings.

Component and Application Process Steps: ectool Example

This sample code shows how to design component and application process steps using the ectool API.

```
#!/bin/bash

# set all names
hostname='localhost'
projectName='default'
appName='myApp'
envName='myEnv'
appTierName='myAppTier'
componentName='myComponent'
```

```
artifactName='DEV:MyArt001'
envTierName='myEnvTier'
artifactProjectName='EC-Artifact-1.0.9.76076'
subProject_artifact='/plugins/EC-Artifact/project'
appProcessName='myApp_process'
appProcessStepName='myApp_process_step'
compProcessName='myComp_process'
compProcessStepName='myComp_process_step'
resource1='res_1'
resource2='res_2'
resource3='res_3'
resource4='res_4'

# login
ectool --server $hostname login admin changeme

# make sure same application, artifacts, resources don't already exist
echo 'make sure same application, artifacts, resources do not already exist'
ectool deleteApplication --projectName $projectName --applicationName $appName
ectool deleteEnvironment --projectName $projectName --environmentName $envName
ectool deleteArtifact --artifactName $artifactName
ectool deleteResource --resourceName $resource1
ectool deleteResource --resourceName $resource2
ectool deleteResource --resourceName $resource3
ectool deleteResource --resourceName $resource4

# create application
echo 'creating application'
ectool createApplication --projectName $projectName --applicationName $appName

#create application tier
echo 'creating application tier'
ectool createApplicationTier --projectName $projectName --applicationName $appName --a
pplicationTierName $appTierName
```

```
#create and publish artifact versions

ectool createArtifact --groupId 'DEV' --artifactKey 'MyArt001'

ectool publishArtifactVersion --artifactName $artifactName --version '1.0' --fromDirectory 'tmp/partial_deployment' --includePatterns "abc1.war"

ectool publishArtifactVersion --artifactName $artifactName --version '2.0' --fromDirectory 'tmp/partial_deployment' --includePatterns "abc2.war"


#create component and add it to app tier

ectool createComponent --projectName $projectName --applicationName $appName --componentName $componentName --pluginName $artifactProjectName

ectool addComponentToApplicationTier --projectName $projectName --applicationName $appName --applicationTierName $appTierName --componentName $componentName


#Set component properties

ectool createProperty --projectName $projectName --applicationName $appName --componentName $componentName --propertyName 'ec_content_details' --propertyType 'sheet'

ectool createProperty --projectName $projectName --applicationName $appName --componentName $componentName --propertyName 'ec_content_details/artifactName' --value $artifactName

ectool createProperty --projectName $projectName --applicationName $appName --componentName $componentName --propertyName 'ec_content_details/versionRange' --value '1.0'

ectool createProperty --projectName $projectName --applicationName $appName --componentName $componentName --propertyName 'ec_content_details/artifactVersionLocationProperty' --value '/myJob/retrievedArtifactVersions/${assignedResourceName}'

ectool createProperty --projectName $projectName --applicationName $appName --componentName $componentName --propertyName 'ec_content_details/overwrite' --value 'update'

ectool createProperty --projectName $projectName --applicationName $appName --componentName $componentName --propertyName 'ec_content_details/filterList' --value ''

ectool createProperty --projectName $projectName --applicationName $appName --componentName $componentName --propertyName 'ec_content_details/pluginProcedure' --value 'Retrieve'

ectool createProperty --projectName $projectName --applicationName $appName --componentName $componentName --propertyName 'ec_content_details/pluginProjectName' --value 'EC-Artifact'

ectool createProperty --projectName $projectName --applicationName $appName --componentName $componentName --propertyName 'ec_ui' --propertyType 'sheet'

ectool createProperty --projectName $projectName --applicationName $appName --componentName $componentName --propertyName 'ec_ui/stepType' --value 'operation'


#create component process and step

ectool createProcess --projectName $projectName --processName $compProcessName --componentApplicationName $appName --processType 'DEPLOY' --componentName $componentName
```

```
ectool createProcessStep --projectName $projectName --processName $compProcessName --processStepName $compProcessStepName --componentName $componentName --componentApplicationName $appName --subprocedure 'Retrieve' --subproject $subProject_artifact --processStepType 'component' --includeCompParameterRef 1
```

```
#create environment, tiers and add resources
```

```
ectool createEnvironment --projectName $projectName --environmentName $envName --environmentEnabled 'true'
```

```
ectool createEnvironmentTier --projectName $projectName --environmentName $envName --environmentTierName $envTierName
```

```
ectool createResource --resourceName $resource1 --hostName $hostname --pools $projectName
```

```
ectool createResource --resourceName $resource2 --hostName $hostname --pools $projectName
```

```
ectool createResource --resourceName $resource3 --hostName $hostname --pools $projectName
```

```
ectool createResource --resourceName $resource4 --hostName $hostname --pools $projectName
```

```
ectool addResourceToEnvironmentTier --resourceName $resource1 --projectName $projectName --environmentName $envName --environmentTierName $envTierName
```

```
ectool addResourceToEnvironmentTier --resourceName $resource2 --projectName $projectName --environmentName $envName --environmentTierName $envTierName
```

```
ectool addResourceToEnvironmentTier --resourceName $resource3 --projectName $projectName --environmentName $envName --environmentTierName $envTierName
```

```
ectool addResourceToEnvironmentTier --resourceName $resource4 --projectName $projectName --environmentName $envName --environmentTierName $envTierName
```

```
# create application process and steps
```

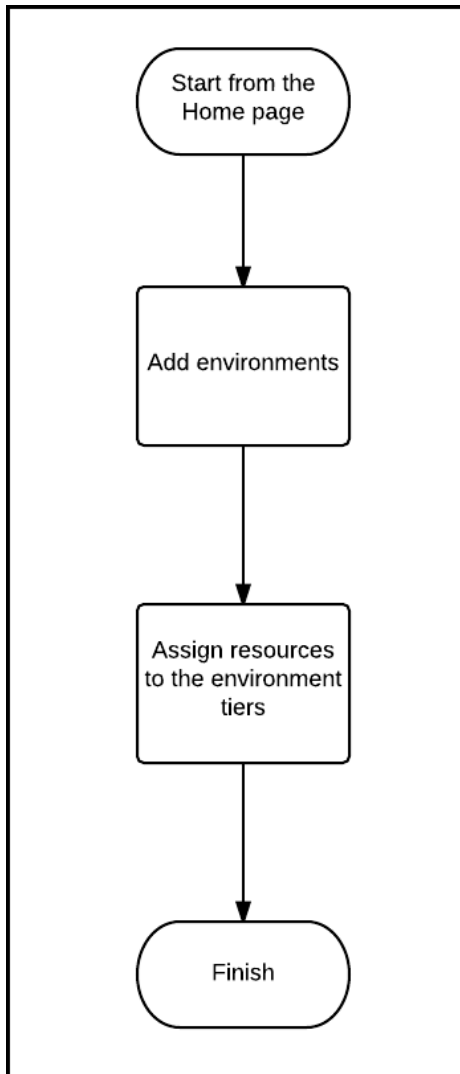
```
ectool createProcess --projectName $projectName --processName $appProcessName --applicationName $appName
```

```
ectool createProcessStep --projectName $projectName --processName $appProcessName --processStepName $appProcessStepName --errorHandling 'abortJob' --subcomponent $componentName --subcomponentApplicationName $appName --subcomponentProcess $compProcessName --applicationName $appName --applicationTierName $appTierName --processStepType 'process'
```

```
# create tier mapping
```

```
ectool createTierMap --projectName $projectName --applicationName $appName --environmentProjectName $projectName --environmentName $envName --tierMapName 'map1' --tierMapping "$appTierName=$envTierName"
```

Modeling Environments



1. Add an environment and environment tiers.
2. Assign resources to environment tiers.

Adding an Environment

Starting from the Home page:

1. Go to the Environments List by either
 - Clicking the **Environments** launch pad.
 - Clicking the **Menu** button > **Environments**.

The Environments List opens.

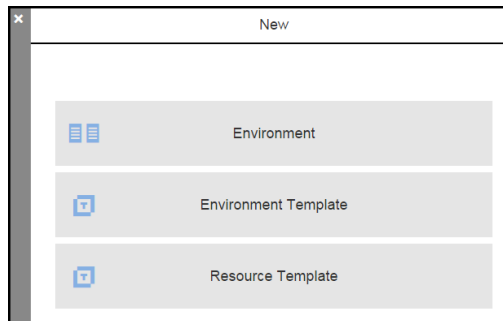
2. Click the **Add +** button in the upper right corner.

Example:



The **New** dialog box appears.

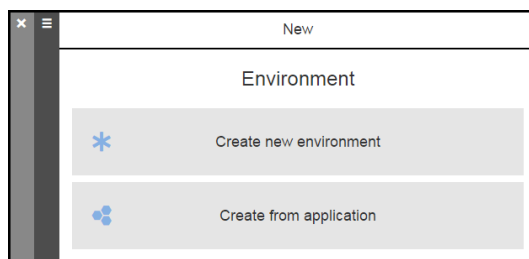
Example:



3. Click **Environment** to create a static environment.

The **New Environment** dialog box appears.

Example:



There are two ways to add an environment:

- Click **Create new environment** to create a environment. Go to the next step.
- Click **Create from application** to create an environment based on an application.

ElectricFlow adds an environment that has the tiers with the same names as the application you select.

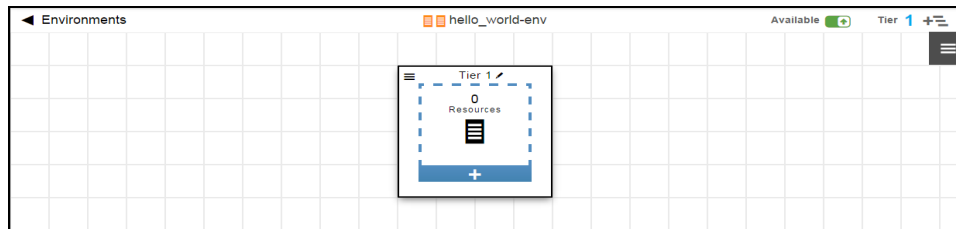
4. Click **Create new environment**. The New dialog box opens..
5. Enter a name in the **Name** field.

6. (Optional) Enter a description of the environment in the **Description** field.
7. Click **OK**.

The Environments Visual Editor opens.

If you are modeling a new environment, the Environments Visual Editor displays an environment tier called Tier 1 with no assigned resources.

Example:



Assigning Resources to Environment Tiers

Starting in the Environments Visual Editor:

1. Click the **Edit** button.
The **Environment Tier Details** dialog box opens.
2. Change the name of the tier and click **OK**.

Example:

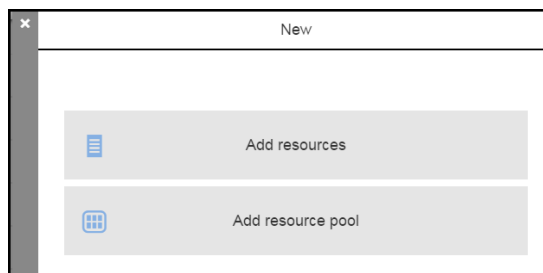
Change the name to **Apache** and click **OK**.

The Applications Visual Editor now has an application tier called hello_world-env.

3. Click the **+** button in the new environment tier to add a resource.

The **New** dialog box opens.

Example:

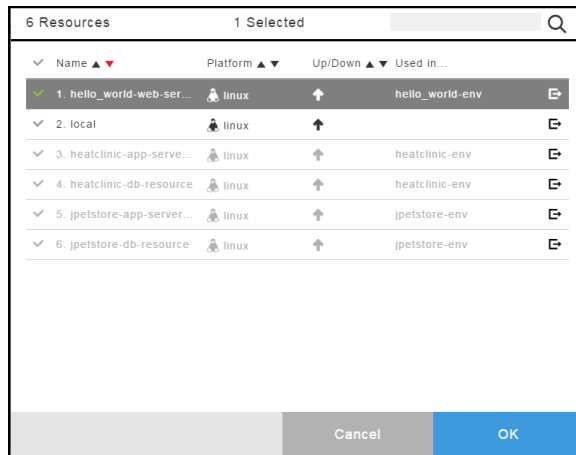


4. Click **Add resources**.

The resource list appears.

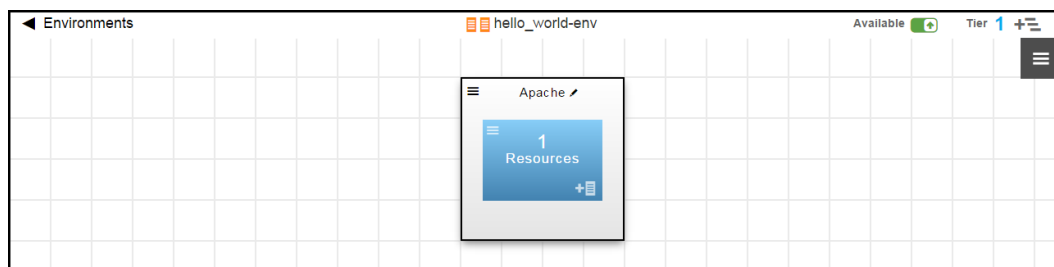
5. Select an available resource and click **OK**.

Example:



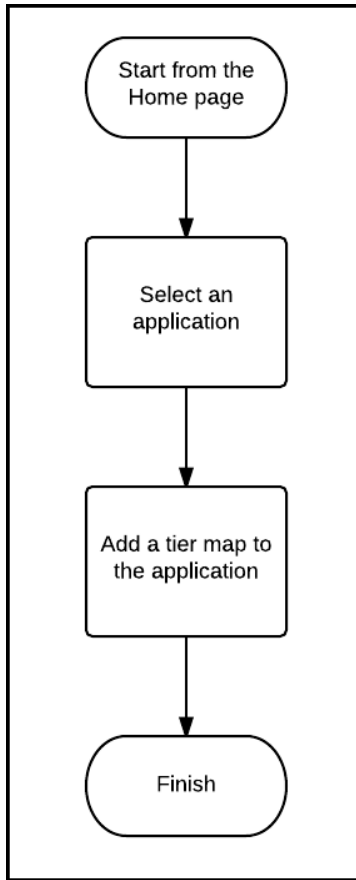
The Environments Visual Editor now shows an environment tier called Apache with one resource.

Example:



6. To add another resource to the same tier, click the **Add resource** button and repeat the previous steps starting with Step 3.

Making Tier Maps



Starting in the Home page:

1. Go to the Applications List.
2. Select the application that you want to run.
The Applications Visual Editor opens.
3. Click the **Add tier map** button.

Example:

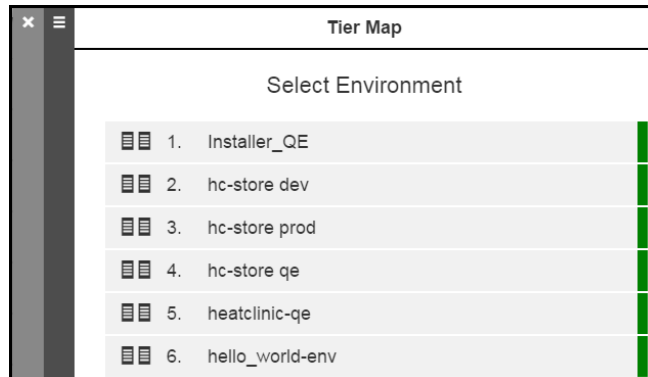


The **Tier Map** dialog box opens.

4. Select an environment to which you want to map the application.

Example:

Select **hello_world-env**.

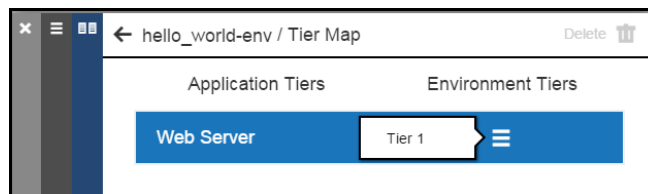


The **<Environment Name>/Tier Map** dialog box opens.

5. Choose an application tier, and click on the button in the Environment Tiers column.

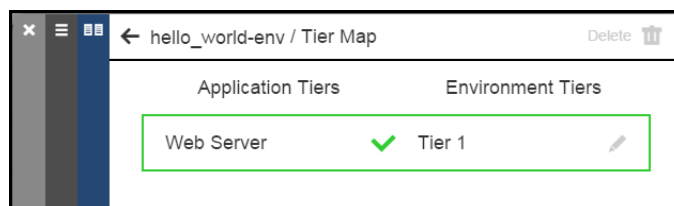
A list of environment tiers in the environment that you selected appears.

Example:



6. Select a tier.
7. If an application has more than one application tier, repeat the previous steps to map the application tiers to environment tiers
8. After you map all the application tiers to environment tiers, click **OK**.

Example:

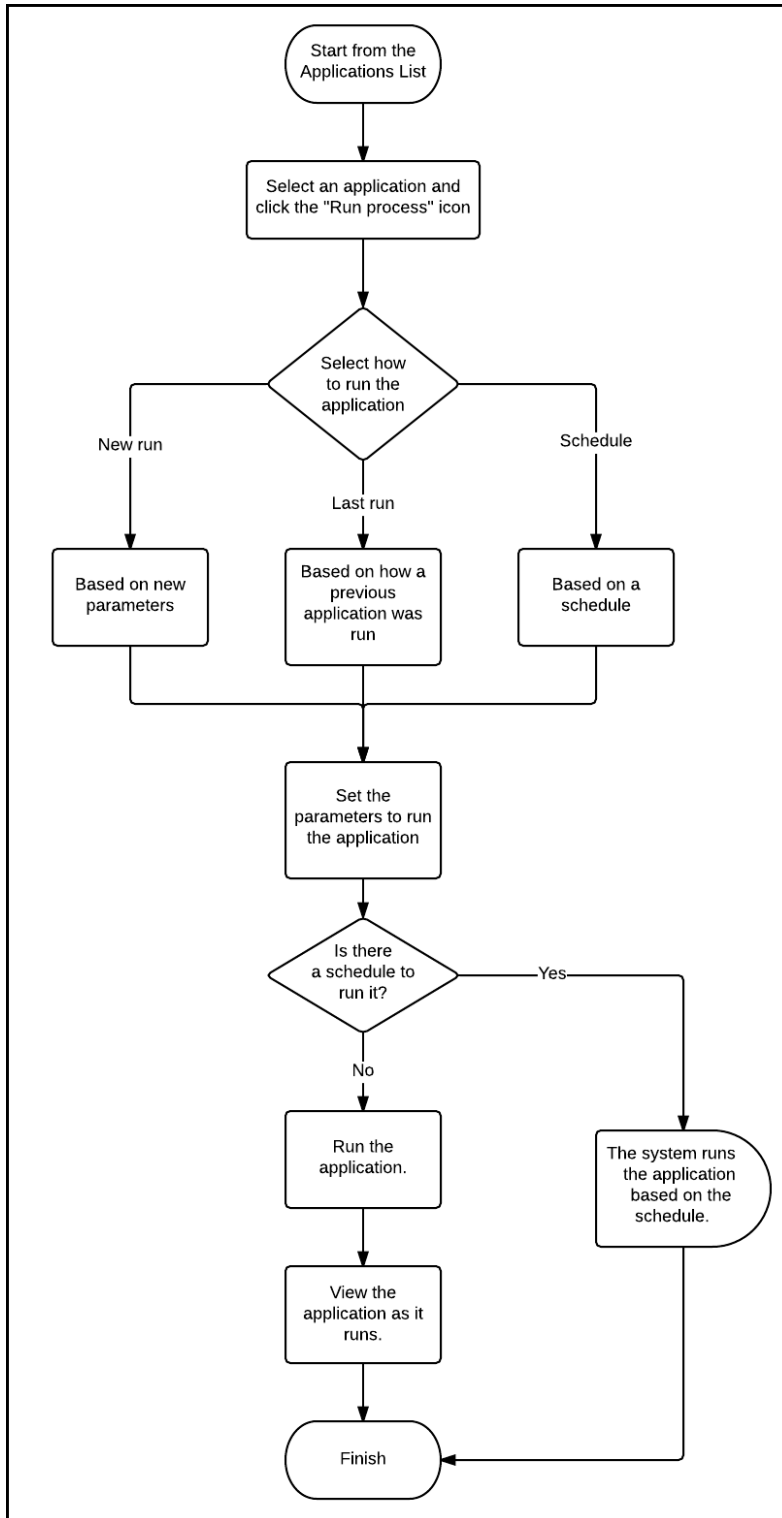


To verify that the application is ready to run, go to the Applications List. When the **Run process** button is green, you can deploy (run) the application.

Deploying (Running) Applications

Note: Within ElectricFlow, the terms *deploy* and *run* are synonymous. When deploying an application in ElectricFlow, you are actually running it to produce your software or application.

Overview



You can deploy (run) an application one of these ways:

- Smart deploy

The system deploys the application only with artifacts that have not been deployed to a resource or selected versions of the artifact have not been deployed to new resources since a previous run.

- Full run

The system runs the application with all the application processes, components, and artifacts in the application.

- Partial run

The system runs the application with only the selected application processes, components, and artifacts in the application.

- Selecting artifacts with specific versions to run

The system runs the application with only the selected versions of the artifacts.

- Snapshot

The system deploys a snapshot of the application.

- A combination of the previous ways.

These are possible combinations to deploy an application:

	Smart Deploy	Full Run	Partial Run	Artifacts with Specific Versions	Snapshot
Smart Deploy		No	Yes	Yes	Yes
Full Run	No		No	Yes	Yes
Partial Run	Yes	No		Yes	Yes
Artifacts with Specific Versions	Yes	Yes	Yes		Yes
Snapshot	Yes	Yes	Yes	Yes	n/a

You can view the results in the Application Inventory and the Environment Inventory.

Deploying an Application

Starting in the Home page:

1. Go to the Applications List.

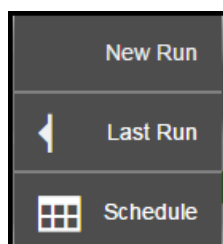
Example:

4 Applications					Select	All	Delete	Add
1	heatclinic-app	1 Component	2 Application Process	1 Tier Map				
2	hello_world-app	1 Component	2 Application Process	1 Tier Map				
3	jpetstore-app	2 Component	2 Application Process	1 Tier Map				
4	sample-app	0 Component	0 Application Process	0 Tier Map				

- Choose an application and click the **Run process** button.

A menu appears.

Example:



If this is the first time that you are running the application, the menu has only the **New Run** and **Schedule** options.

- To specify how you want to deploy (run) the application, select one of these options:
 - New run**—Set the parameters as described in [Running Applications with New Parameters](#) .
The **New** <Application name> dialog box opens.
 - Last run**—Use the parameters from a previous run. You can modify one or more of these parameters as described in [Running Applications with Parameters from Previous Runs](#).
The **Edit** <Application name> dialog box opens.
 - Schedule**—Set the application to run on a schedule as described in [Running Applications with Schedules](#).

Running Applications with New Parameters

Starting on the Home page:

- Go to the Applications List.

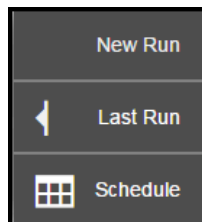
Example:

4 Applications					Select	All	Delete	Add
1	heatclinic-app	1 Component	2 Application Process	1 Tier Map				
2	hello_world-app	1 Component	2 Application Process	1 Tier Map				
3	jpetstore-app	2 Component	2 Application Process	1 Tier Map				
4	sample-app	0 Component	0 Application Process	0 Tier Map				

2. Choose an application and click the **Run process** button.

A menu appears.

Example:



If this is the first time that you are running the application, the menu has only the **New Run** and **Schedule** options.

3. Select **New Run** to deploy the application with new parameters.

The **New** dialog box to set the parameters to deploy the application opens.

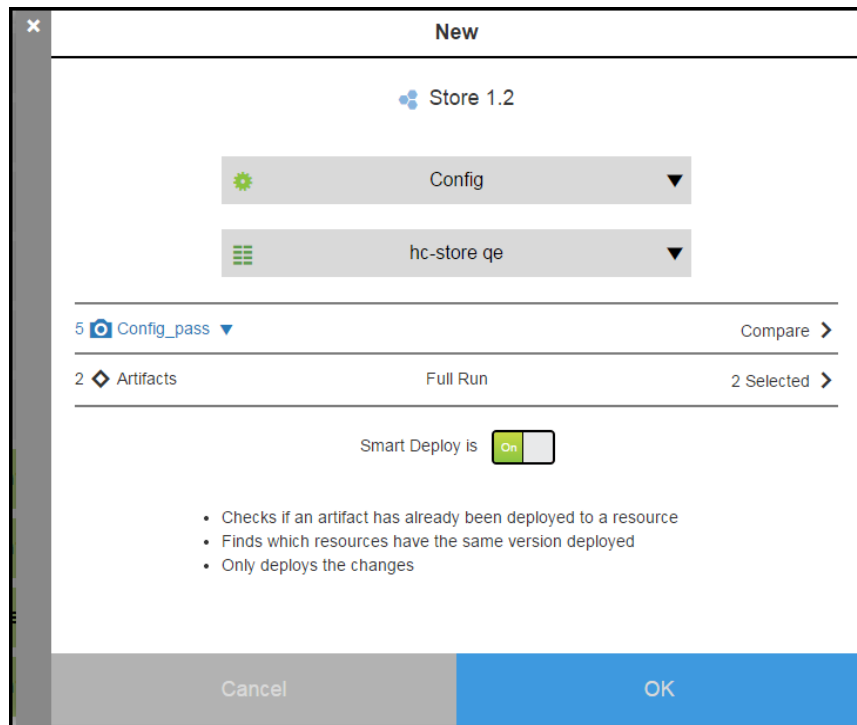
4. Select the application process.
5. Select an environment.

6. (Optional) Select a snapshot to deploy.

Note: If this is the first time that you are running the application, smart deploy is not enabled.

Example:

This example is not part of the hello_world-env deployment. It shows only that the *Config* process, which is mapped to the environment called hc-store qe, has five snapshots.

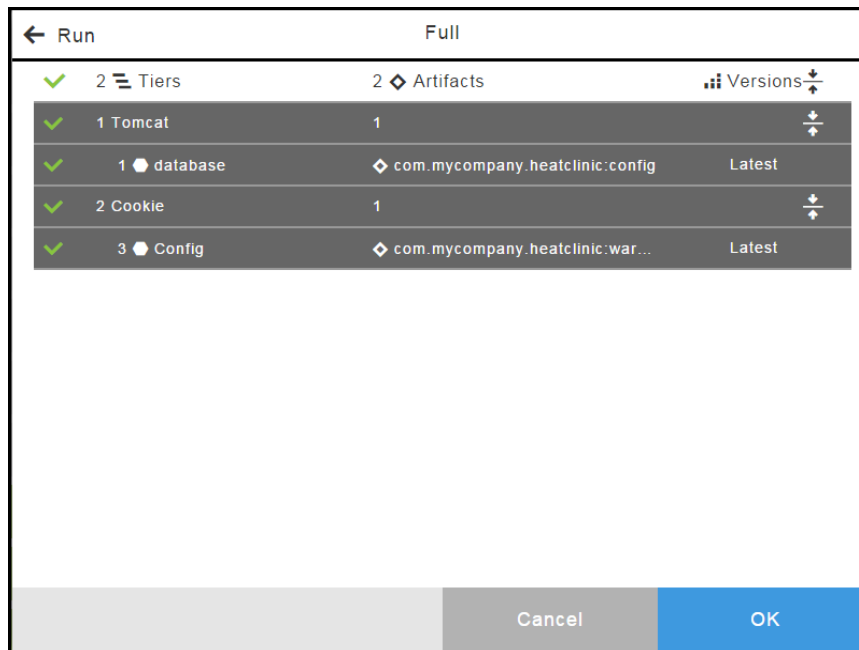


7. (Optional) To select and configure the **Full Run** option, click **Full Run**, and then click **OK**.

A dialog box opens showing the objects in the application.

Example:

This example is not part of the hello_world-env deployment. It shows only that all of the objects, application tiers, components, and artifacts, are selected.



8. (Optional) To select and configure the **Partial Run** option:

a. Click **Partial Run**.

A dialog box opens showing the objects in the application.

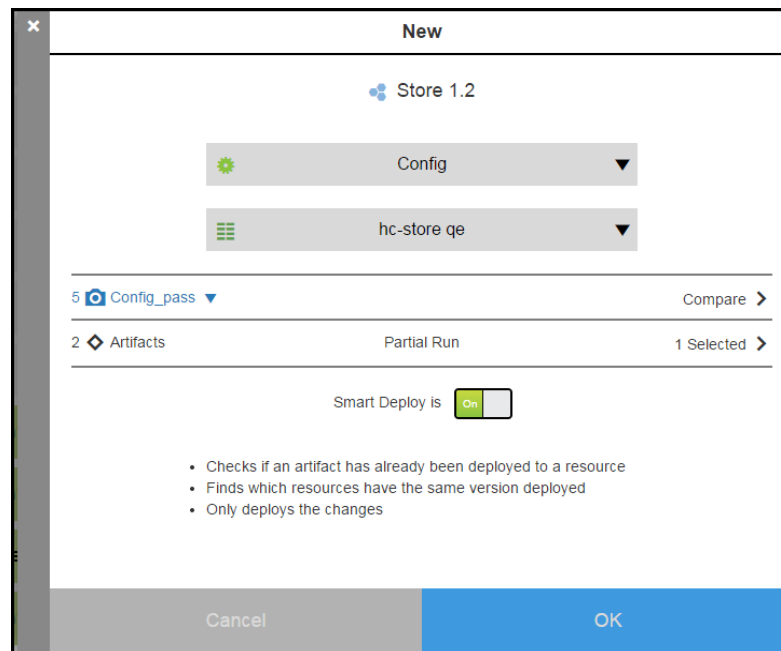
b. Determine the objects in the application that you do not want to run, and click the each row to remove them from the run.

c. Click **OK**.

The dialog box now shows that the **Partial Run** is selected and that only one of two artifacts in application will run.

Example:

This example is not part of the hello_world-env deployment. It shows only that **Partial Run** is selected.



9. (Optional) To select artifacts with specific versions:

- a. Click **Selected Artifacts**.

A dialog box opens showing the objects in the application. The version of each component is in the Version column. The current version of all the components is *Latest*.

Example:

This example is not part of the hello_world-env deployment. It shows only that two objects in the tier will run.



- b. To change the version of a component, click the down arrow next to current version.

A drop-down menu appears.

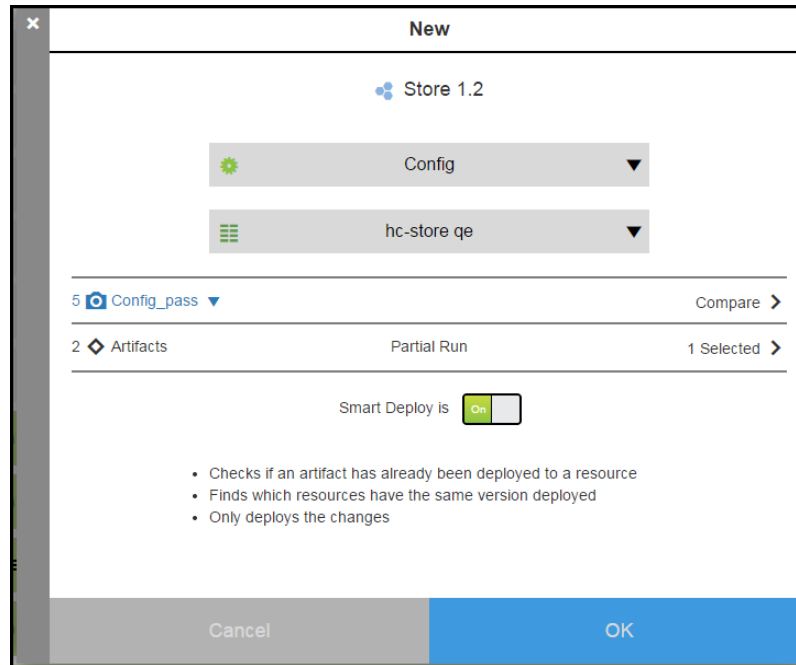
- c. Select the version that you want the application to run.

d. Click **OK**.

The dialog box re-appears.

Example:

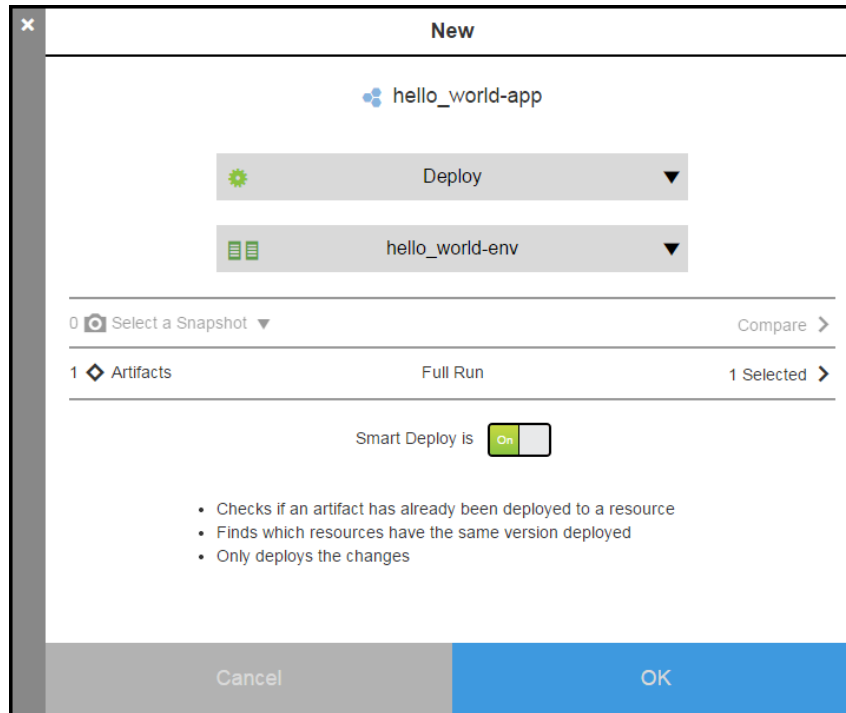
This example is not part of the hello_world-env deployment. It shows only that the settings for the Partial Run.



- Click **OK** to run the application.

Example:

This example shows one way to deploy the **hello_world-app** application.



You can view the results in the Application Inventory and the Environment Inventory. For more information, go to [Viewing Results and Troubleshooting](#).

Deploying Applications with Parameters from Previous Runs

When you use the **Last Run** option, you configure how to run the application based on parameters from a previous run.

- Go to the Applications List.

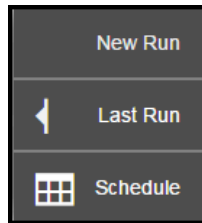
Example:

Applications					Select	All	Delete	Add
1	heatclinic-app	1 Component	2 Application Process	1 Tier Map				
2	hello_world-app	1 Component	2 Application Process	1 Tier Map				
3	jpetstore-app	2 Component	2 Application Process	1 Tier Map				
4	sample-app	0 Component	0 Application Process	0 Tier Map				

2. Choose an application and click the **Run process** button.

A menu appears.

Example:



3. Select **Last Run** to use the parameters from previously run application.

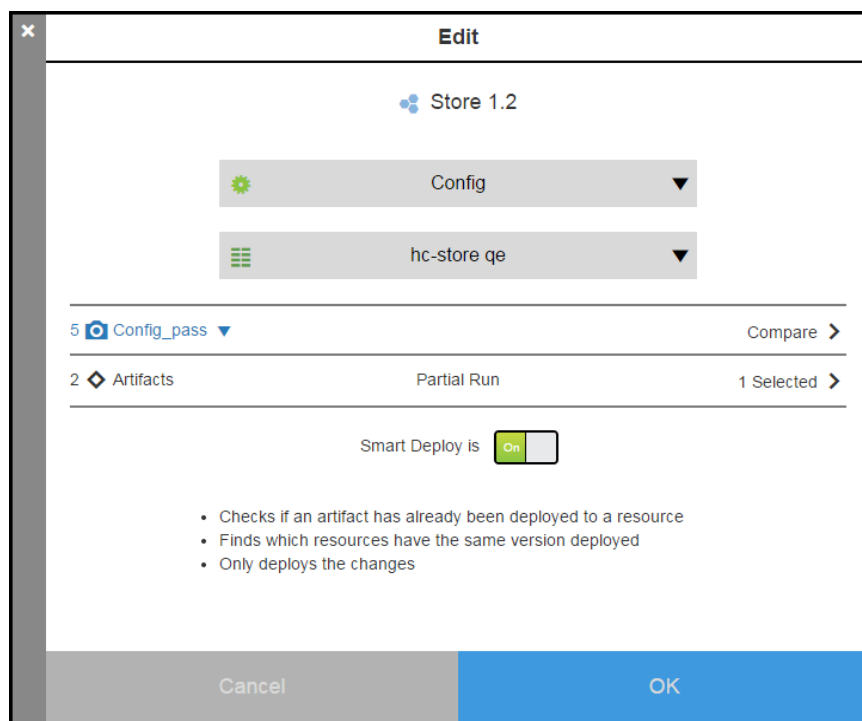
A list of applications appears.

4. Select a previous run.

The dialog box to set the parameters for running the application opens.

Example:

This example is not part of the hello_world-env deployment. It shows only the deployment for Store 1.2 is a Partial Run.



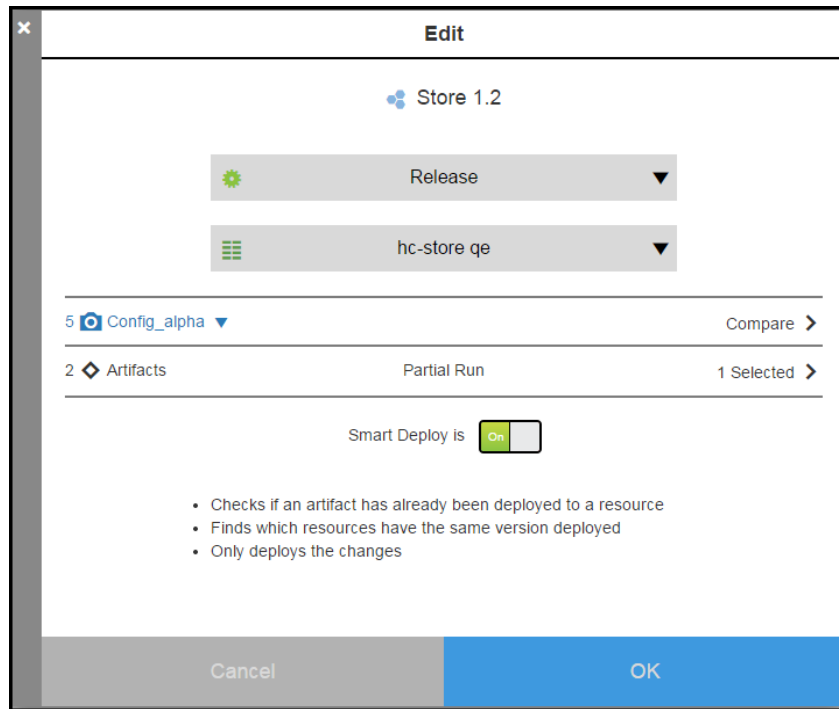
5. Select the application process.

6. Select an environment.
7. (Optional) Select a snapshot.

If this is the first time that you are running the application, smart deploy is not enabled.

Example:

This example is not part of the hello_world-env deployment. It shows only that there are five snapshots.



8. To select and configure the **Full Run** option:

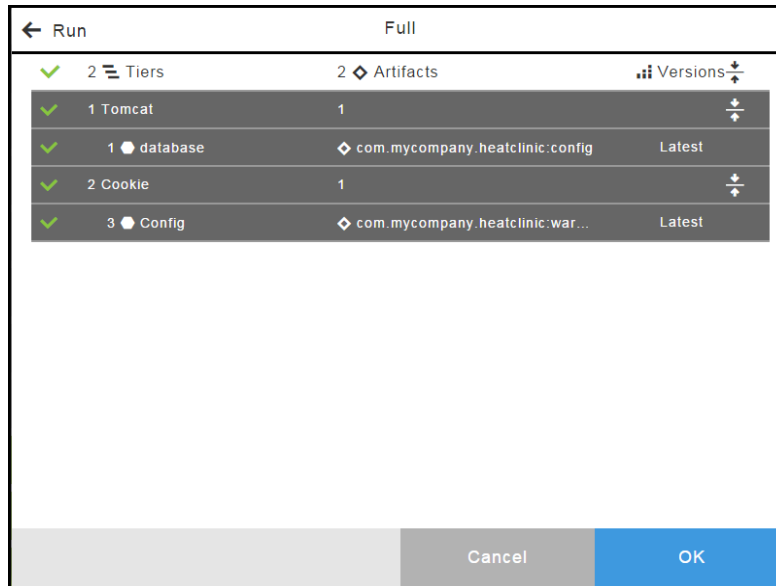
- a. Click **Full Run**.

A dialog box opens showing the objects in the application.

This example shows that all of the objects, application tiers, components, and artifacts, are selected.

Example:

This example is not part of the hello_world-env deployment. It shows only that the all objects are selected for the Full Run.

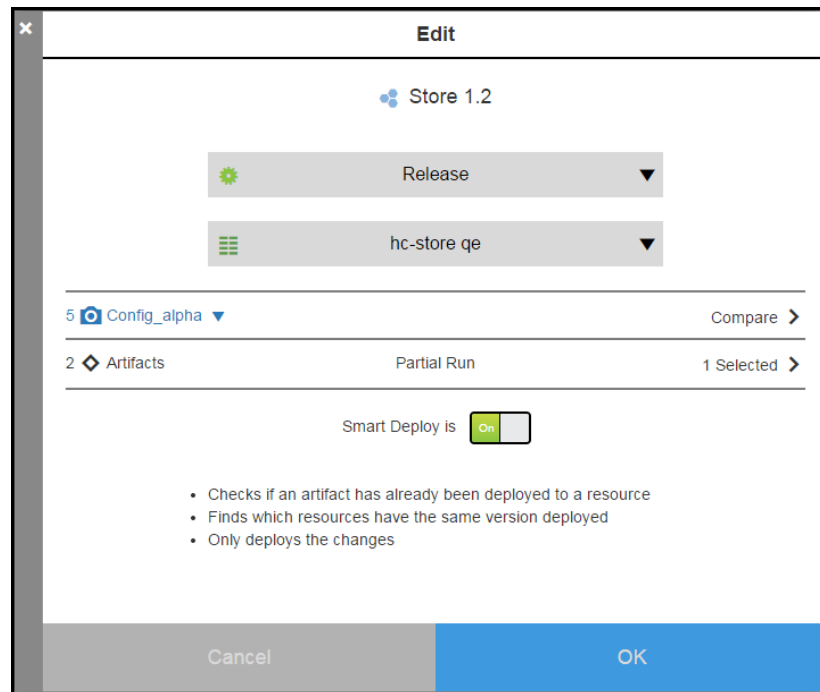


- b. Click **OK**.

The dialog box re-appears.

Example:

This example is not part of the hello_world-env deployment. It shows only that one of two artifacts will be deployed.



9. To select and configure the **Partial Run** option:

- a. Click **Partial Run**.

A dialog box opens showing the objects in the application.

- b. Determine the objects in the application that you do not want to run and click the each row to remove them from the run.
- c. Click **OK**.

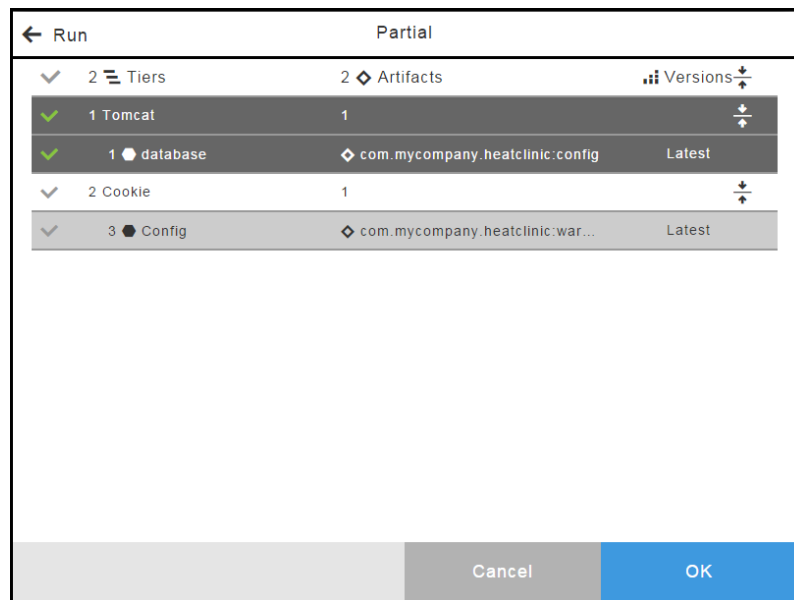
10. To select artifacts with specific versions:

- a. Click **Selected Artifacts**.

A dialog box opens showing the objects in the application. The version of each component is in the Version column. The current version of all the components is *Latest*, the latest version of the component.

Example:

This example is not part of the hello_world-env deployment. It shows how to select specific artifacts to deploy.



- b. To change the version of a component, click the down arrow next to current version.

A drop-down menu appears.

- c. Select the version that you want the application to run.

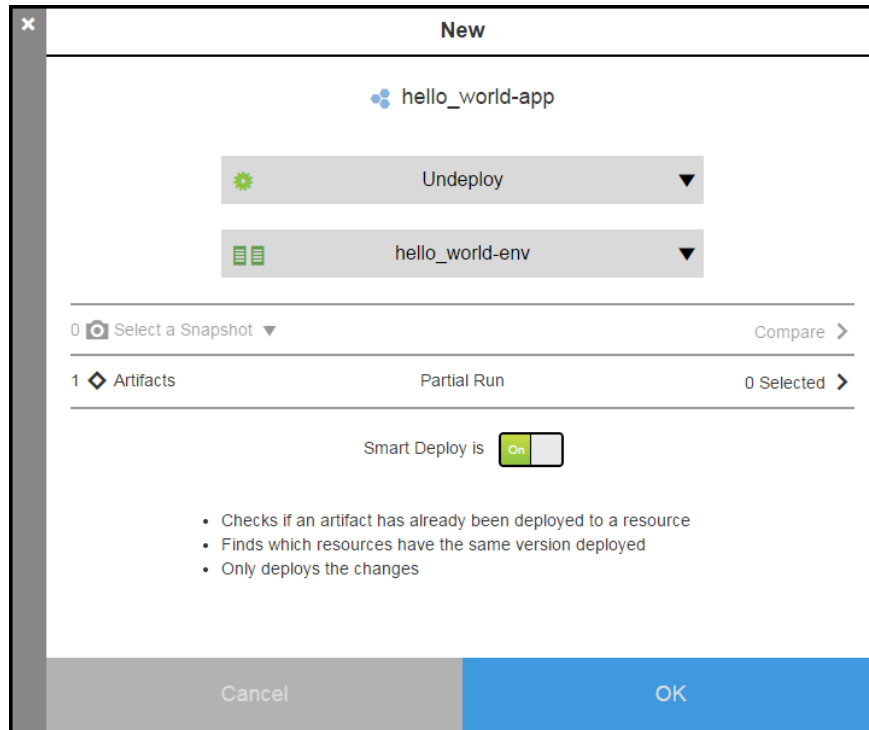
- d. Click **OK**.

The dialog box re-appears.

- Click **OK** to run the application.

Example:

This example shows one way to deploy the **hello_world-app** application.



You can view the results in the Application Inventory and the Environment Inventory. For more information, go to [Viewing Results and Troubleshooting](#).

Deploying (Running) Applications with Schedules

When you use the **Schedule** option, you set the day and time when the application runs.

- Go to the Applications List.

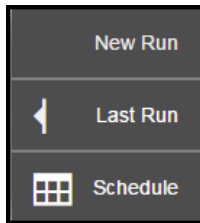
Example:

Applications					Select	All	Delete	Add
1	heatclinic-app	1 Component	2 Application Process	1 Tier Map				
2	hello_world-app	1 Component	2 Application Process	1 Tier Map				
3	jpetstore-app	2 Component	2 Application Process	1 Tier Map				
4	sample-app	0 Component	0 Application Process	0 Tier Map				

2. Choose an application and click the **Run process** button.

A menu appears.

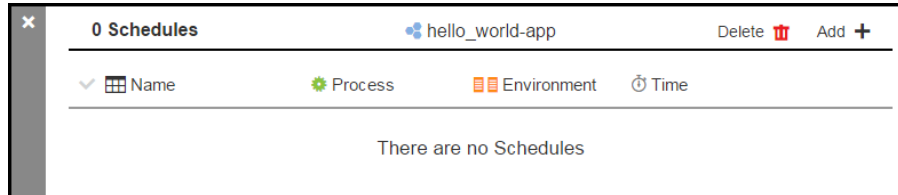
Example:



3. Select **Schedule** to set the day and time when the application will run automatically.

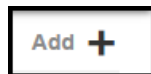
The **Schedule** dialog box opens.

Example:



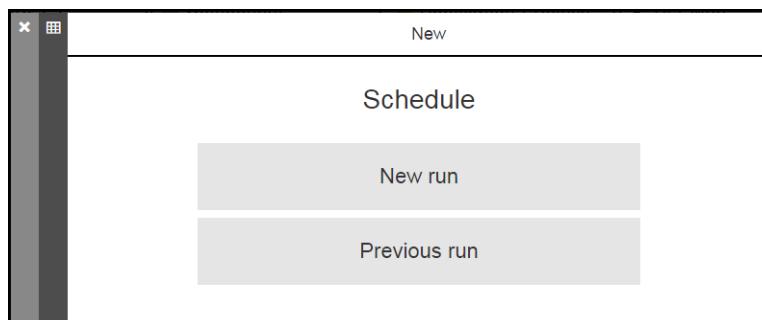
- a. Click the **Add +** button in the upper right corner.

Example:



The **New Schedule** dialog box appears.

Example:



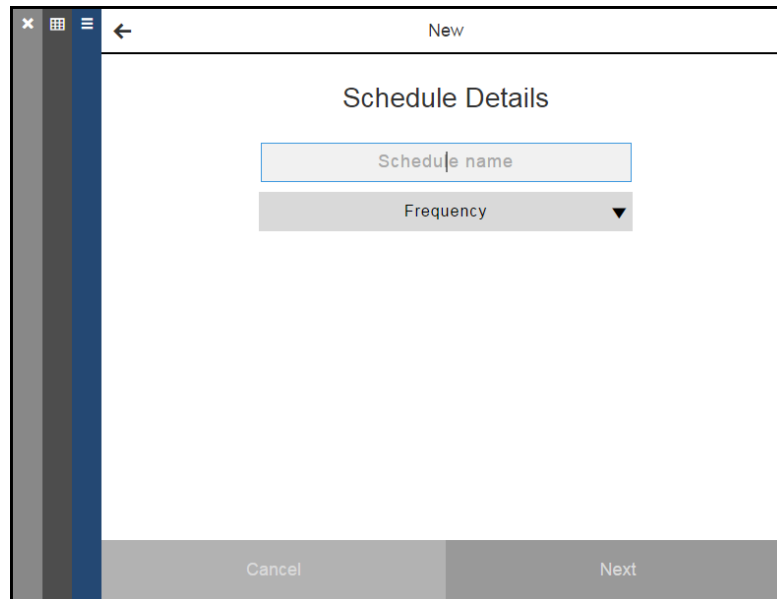
- b. Select **New run** or **Previous run**.

If this is the first time that you are running the application, select **New run** and go to the appropriate next step.

If you select **Previous Run**, go to the appropriate next step.

- c. Select **New run** to set a schedule for a new run, and enter a name for the schedule in the **New Schedule Details** dialog box.

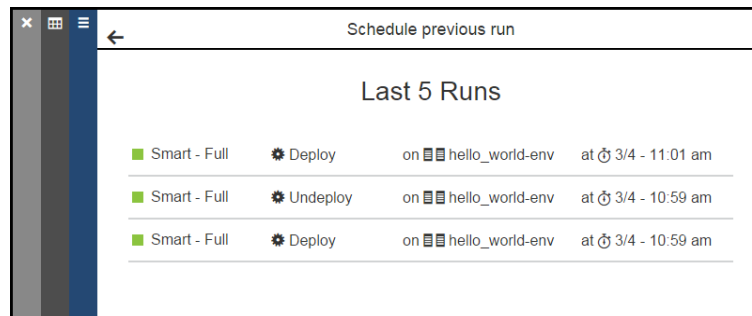
Example:



The screenshot shows a dialog box titled "New" with a back arrow. Inside, the title "Schedule Details" is centered. Below the title are two input fields: "Schedule name" and "Frequency" with a dropdown arrow. At the bottom are "Cancel" and "Next" buttons.

- d. Select **Previous run** to set a schedule used for a previous run, and select it in the **Schedule previous run** dialog box, which has information about the last five runs.

Example:



The screenshot shows a dialog box titled "Schedule previous run" with a back arrow. Inside, the title "Last 5 Runs" is centered. Below the title is a table with three rows of run data.

Last 5 Runs			
Smart - Full	Deploy	on hello_world-env	at 3/4 - 11:01 am
Smart - Full	Undeploy	on hello_world-env	at 3/4 - 10:59 am
Smart - Full	Deploy	on hello_world-env	at 3/4 - 10:59 am

The **New Schedule Details** dialog box opens.

- e. Enter the schedule name and click in the **Frequency** field to set how often the application runs.

Example:

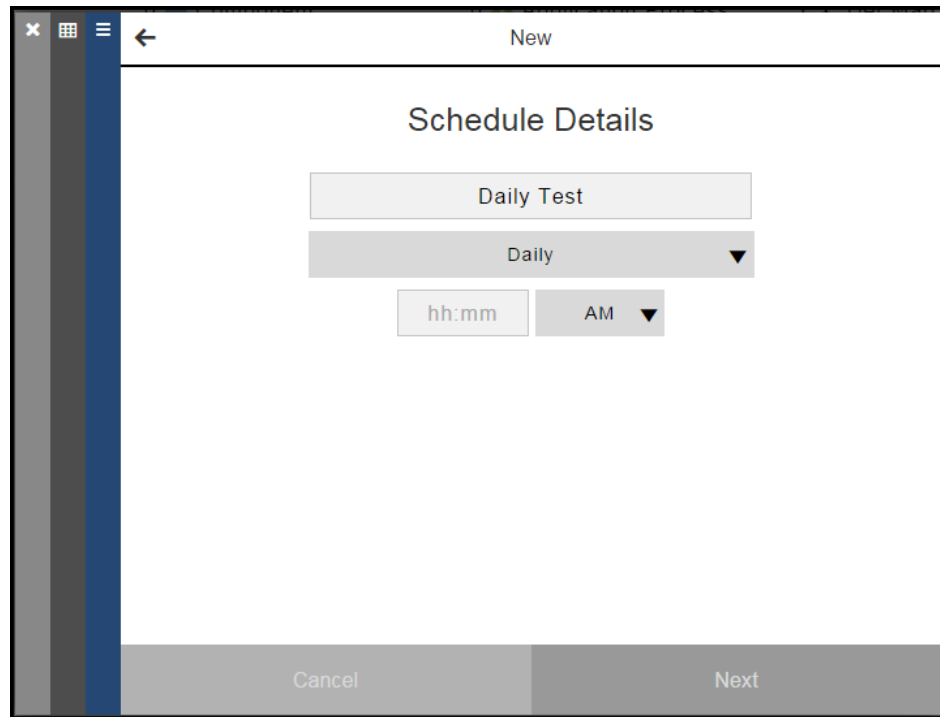
The screenshot shows a mobile application interface for creating a new schedule. The top bar is labeled 'New' with a back arrow. The main section is titled 'Schedule Details'. It contains a text field with the value 'Daily Test'. Below this is a 'Frequency' dropdown menu, which is currently open, displaying the following options: 'Once', 'Daily', 'Weekly', and 'Monthly'. At the bottom of the screen, there are two buttons: 'Cancel' and 'Next'.

Depending on the frequency that you select, different fields appear.

- **Once**—Enter information in the date (Month, Day, and Year) and the time (hours, minutes, and **AM** or **PM**) fields.
- **Daily**—Enter only the time (hours, minutes, and **AM** or **PM**).
- **Weekly**—Enter the **Day of week** (Monday to Sunday) and the time (hours, minutes, and **AM** or **PM**).
- **Monthly**—Enter the **Day** and the time (hours, minutes, and **AM** or **PM**).

- f. Enter the appropriate information in the fields below the **Frequency** field.

Example:



The screenshot shows a mobile application interface for scheduling. At the top, there is a header bar with a back arrow, a 'New' title, and a close button. Below the header, the title 'Schedule Details' is centered. The form contains three main input fields: a text field with 'Daily Test', a dropdown menu with 'Daily', and a time selection field with 'hh:mm' and 'AM'. At the bottom, there are two buttons: 'Cancel' and 'Next'.

For example, if you select **Daily**, set the time (hours, minutes, and **AM** or **PM**). In the time fields, click in it and enter the hours and minutes and then select **AM** or **PM**.

Example:

The screenshot shows a 'Schedule Details' dialog box. The title bar includes a 'New' button and a back arrow. The main area is titled 'Schedule Details'. It features a text input field containing 'Daily Test', a frequency dropdown menu currently set to 'Daily', a time input field showing '01:__', and a time period dropdown menu with 'AM' selected. The bottom of the dialog has two buttons: 'Cancel' and 'Next'.

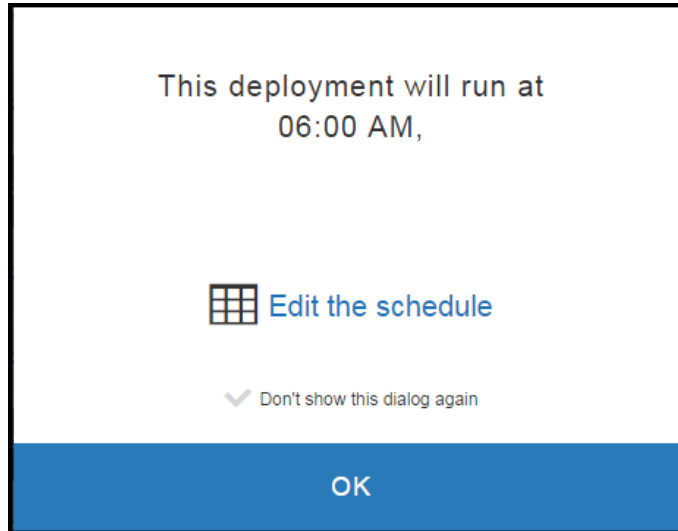
- g. Click **Next**. The dialog box to set the parameters for running the application opens.
4. Select an application process.
 5. Select an environment.

If this is the first time that you are running the application, smart deploy is not enabled.

6. Click **OK**.

A message appears about when the application is run according to the schedule.

Example:



When it is time for the application to run, the system runs it in the background.

You can view the results in the Application Inventory and the Environment Inventory. For more information, go to [Viewing Results and Troubleshooting](#).

Deploying Snapshots

1. Go to the Applications List.
2. Choose an application.
3. Click the **Run process** button.

Example:

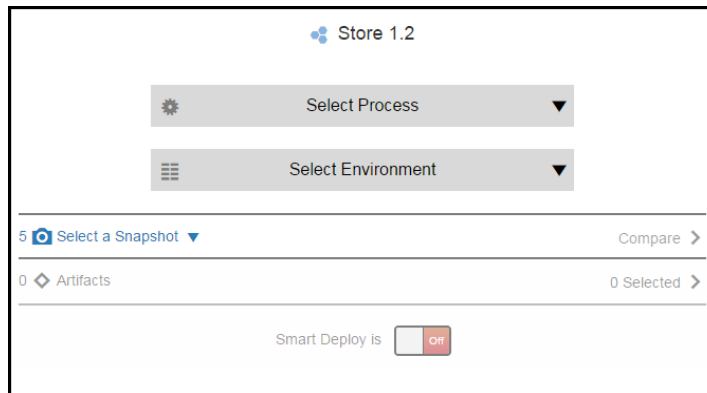


4. Select **New Run**, **Last Run**, or **Schedule**.

The dialog box to set the parameters for running an application opens.

In this dialog box, you can deploy a snapshot or compare the application to the selected snapshot.

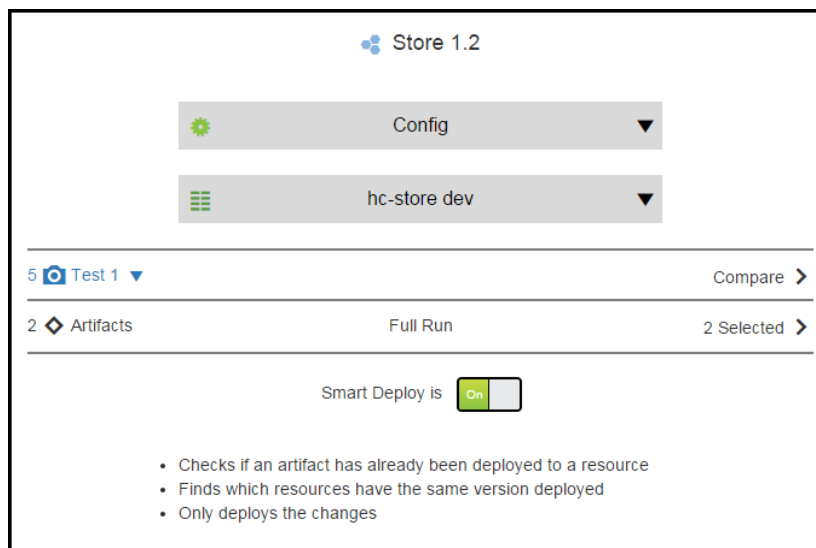
Example:



5. Select the parameters to run an application.

For more information, go to [Deploying or Comparing Snapshots](#) on page 508 and [Deploying \(Running\) Applications](#) on page 152.

Example:



6. To run (deploy) the snapshot:

1. Click **OK**.
2. Go to [Deploying \(Running\) Applications](#) on page 152 for more information.

7. To compare the application to the selected snapshots:

1. Click **Compare** to compare the application to the selected snapshot.
2. Go to [Comparing Snapshots](#) on page 511.

You can compare the application to other snapshots.

Deploying or Comparing Snapshots

How to get here: From the Home page, go to the Applications List, choose an application, click the Run process button, and select **New Run**.

Example:



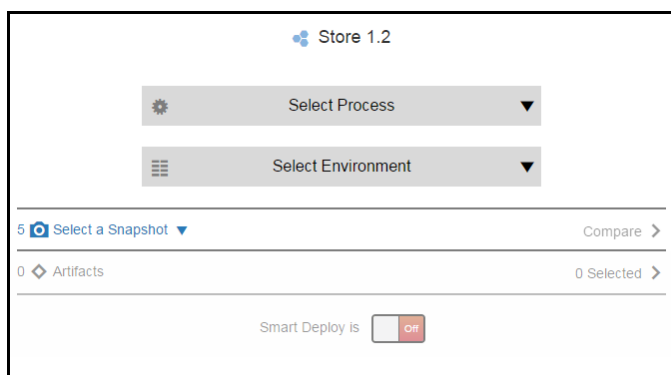
The dialog box to set the parameters for running (deploying) the application opens.

In this dialog box, you can deploy a snapshot or compare the application to a snapshot.

Setting Parameters in the Dialog Box

In the dialog box, the **Select a Snapshot** option is available (enabled) because the application has one or more snapshots saved.

Example:



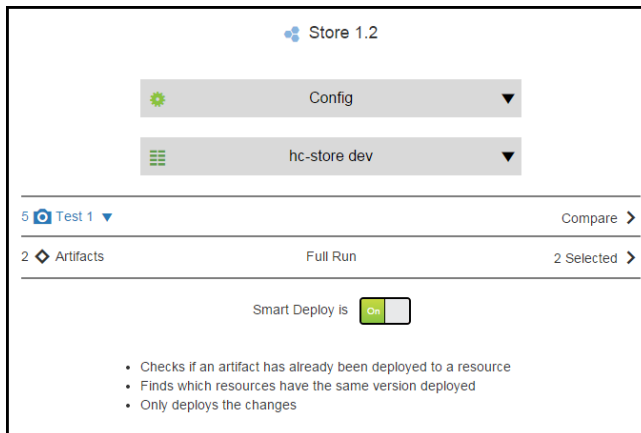
Select the following options to set the parameters to run the application:

- **Select Process**—Click the pull-down button to select the application to run.
- **Select Environment**—Click the pull-down button to select the environment in which the application will run.
- **Select a Snapshot**—Click the pull-down button to select one or more snapshot.

The **Compare** option is now available (enabled) because of the following

- There are one or more snapshots that can be compared to the application in the selected environment.
- The application has been selected.
- The environment has been selected.
- The one or more snapshot has been selected.

Example:



Deploying Snapshots

After setting the parameters, click **OK** to run (deploy) the selected snapshot.

For more information, go to

Comparing Snapshots

After setting the parameters, click **OK** to compare the selected application to a snapshot.

Deploying Applications with Parameters

Starting in the Applications List:

1. Choose the application that you want to deploy and click the **Run process** button.
2. Select **New Run** to deploy the application with new settings.

The **New** dialog box opens.

3. Select the application process, environment, tier map, snapshot, artifact, and resource options, as described in other topics in this document.



4. In the **Parameters** row, click the button to open a form showing the parameters that apply to the application process.
5. Enter information in the fields.

- Click **OK** to save the parameter settings and close the form.

The **New** dialog box now shows what you set in the previous two steps.

The Parameters row shows the number of required parameters. You must enter information for those parameters to deploy the application.

- Click **OK** to deploy the application.

Examples: Deploying (Running) Applications

These examples show how to set the parameters for:

- Full run

The system runs the application with all the application processes, components, and artifacts in the application.

- Partial run

The system runs the application with only the selected application processes, components, and artifacts in the application.

- Selecting artifacts with specific versions to run

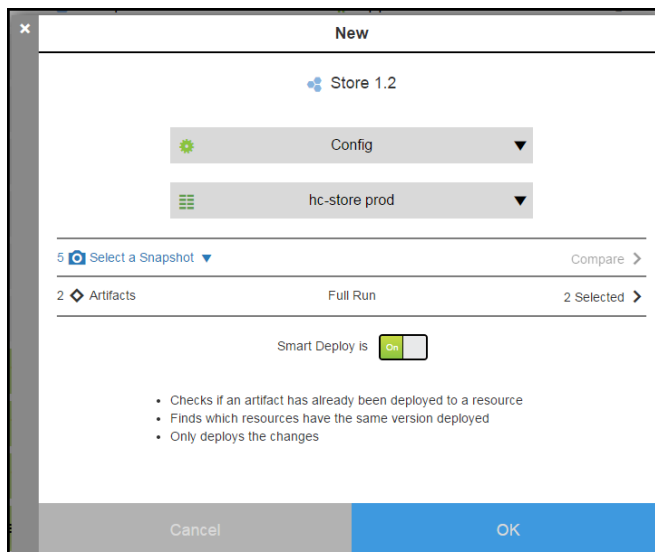
The system runs the application with only the selected versions of the artifacts.

- Combinations of these ways

Full Run

In this example, **Full Run** is selected. All the objects in Store 1.2 will be deployed.

Examples:



To view what objects in the application will run, click **Full Run**.

A dialog box with a list of objects in the application opens. The objects that are selected with the green check mark will run.

Example:

← Run Full		
✓ 2 Tiers	2 Artifacts	Versions
✓ 1 Tomcat	1	
✓ 1 database	com.mycompany.heatclinic.config	Latest
✓ 2 Cookie	1	
✓ 4 Config	com.mycompany.heatclinic.war...	Latest

Partial Runs with Specific Artifact Versions

You can do a **Partial Run** to run only some of the objects.

To select an object that you do not want to deploy, click in its row.

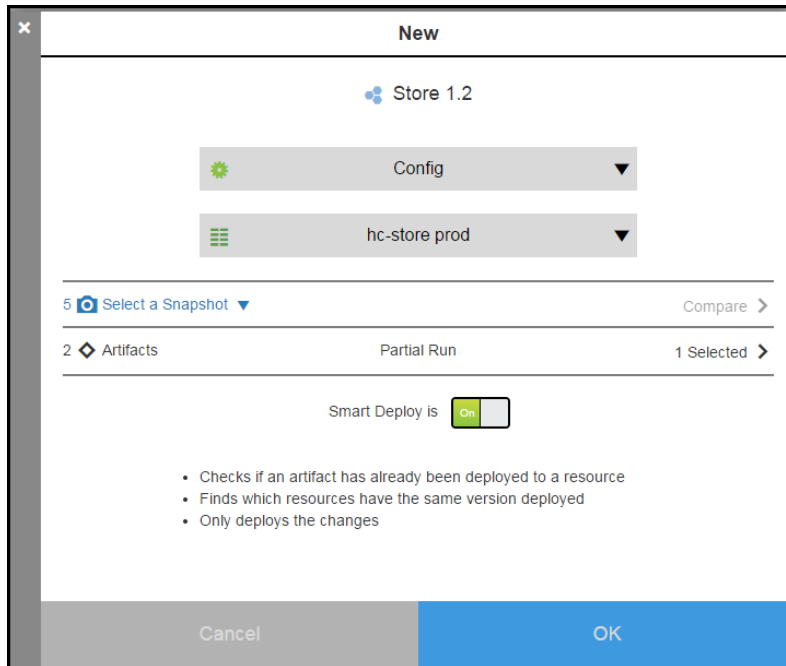
For example, if you click in the Cookie row, all the objects in the Cookie are removed from the next run.

Example:

← Run Partial		
✓ 2 Tiers	2 Artifacts	Versions
✓ 1 Tomcat	1	
✓ 1 database	com.mycompany.heatclinic.config	Latest
2 Cookie	1	
4 Config	com.mycompany.heatclinic.war...	Latest

After you click **OK**, the dialog box now shows that the when the application runs, it is a *Partial Run* and that *1 of 2* artifacts in the application will run.

Example:



You can also select specific versions of artifacts.

To select the artifact versions, click in the row with Artifacts.

The dialog box opens.

Example:

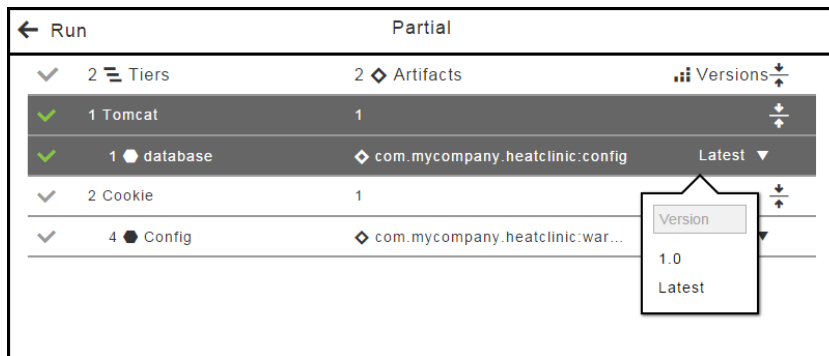
← Run		Partial	
✓ 2 Tiers	2 Artifacts	Versions	
✓ 1 Tomcat	1		
✓ 1 database	com.mycompany.heatclinic:config	Latest	
✓ 2 Cookie	1		
✓ 4 Config	com.mycompany.heatclinic:war...	Latest	

The version of each artifact is in the Version column. The current version is *Latest*.

To see the actual version, click the down arrow next to the current version. A drop-down menu appears.

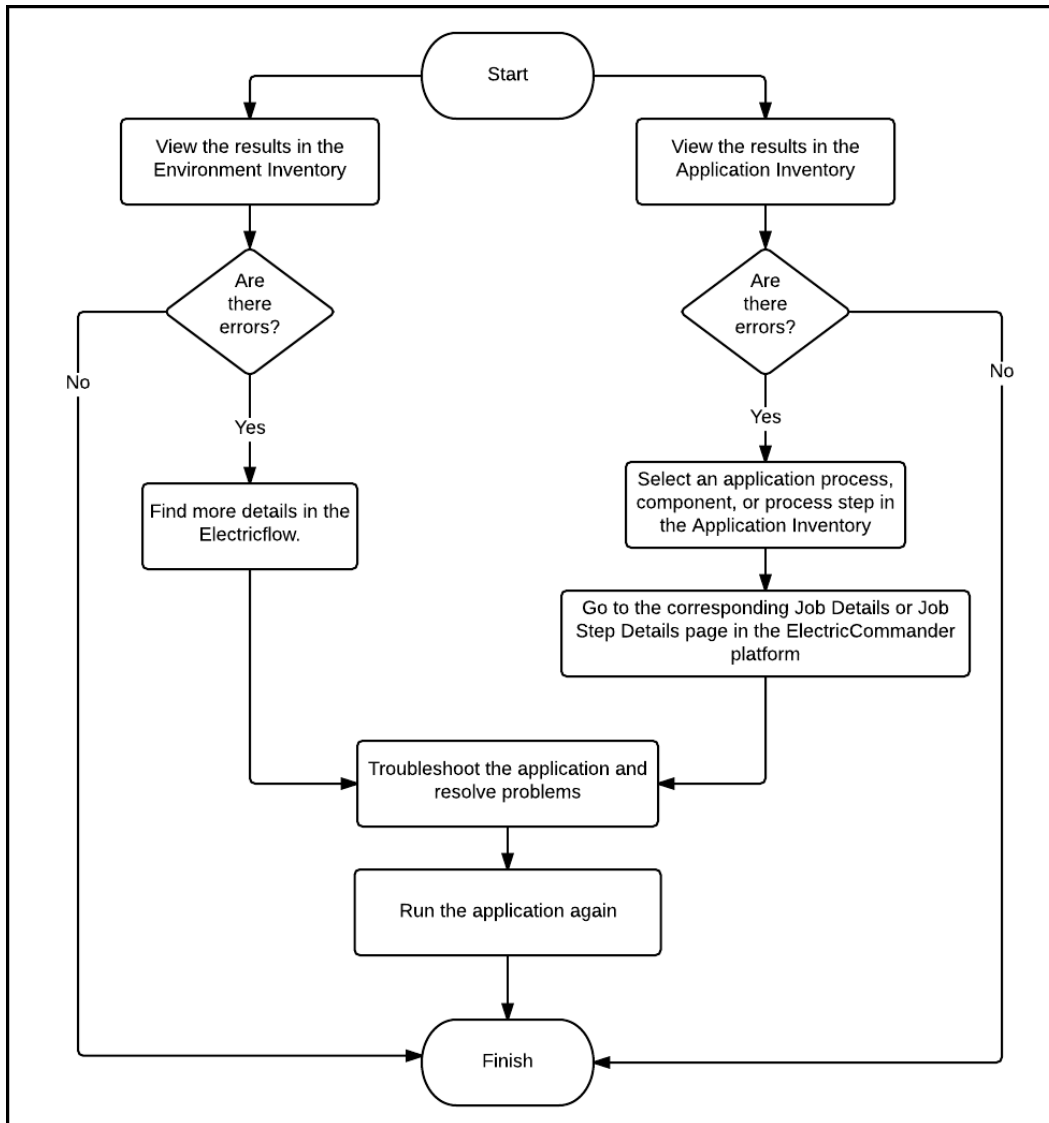
For the Backup1.zip component, the possible versions are in the drop-down list. The latest version is Version 2.0.

Example:



Click **OK** to save these settings and return to the dialog box.

Viewing Results and Troubleshooting



Viewing the Environment Inventory and Application Inventory

Follow these steps to get more information about the state of the environment at a point in time and to view information about an object during the deployment.

Starting at the Home page:

1. Go to the Environments List.
2. Choose an environment.

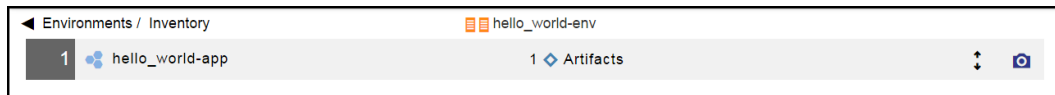
3. Click the **Inventory** button.

Example:



The Environment Inventory opens.

Example:



4. Click the **View** button to view more information about the application.

Example:



Information about the components appears.

Example:



- Click the **Process** button to view more information.

Example:



The Application Inventory opens and displays details about the application processes, components, and process steps.

Example:

Applications / View Run admin Running: hello_world-app - Deploy on hello_world-env Errors 0									
	3_Deploy_hello_world-app_Default_20150304...	★ Success	Mar 04, 2015	11:01 Pac...	00:01	÷	100%	⊕	➔
	put html files	★ Success	Mar 04, 2015	11:01 Pac...	00:01	÷	100%	⊕	➔
	get html files	★ Success	Mar 04, 2015	11:01 Pac...	00:00	÷	100%	⊕	➔
	copy to apache	★ Success	Mar 04, 2015	11:01 Pac...	00:00	÷	100%	⊕	➔

Note: You can also go to the Application Inventory by going to the Applications List, clicking the **View** button to show the previous deployments, choosing a deployment, and clicking the **View Details** button.

- Click the **View** buttons for the process steps to view more details.

Example:

Applications / View Run admin Running: hello_world-app - Deploy on hello_world-env Errors 0									
	3_Deploy_hello_world-app_Default_20150304...	★ Success	Mar 04, 2015	11:01 Pac...	00:01	÷	100%	⊕	➔
	put html files	★ Success	Mar 04, 2015	11:01 Pac...	00:01	÷	100%	⊕	➔
	get html files	★ Success	Mar 04, 2015	11:01 Pac...	00:00	÷	100%	⊕	➔
	hello_world-web-server-resource	★ Success	Mar 04, 2015	11:01 Pac...	00:00	÷	100%	⊕	
	copy to apache	★ Success	Mar 04, 2015	11:01 Pac...	00:00	÷	100%	⊕	➔
	hello_world-web-server-resource	★ Success	Mar 04, 2015	11:01 Pac...	00:00	÷	100%	⊕	

Examples: Viewing Deployment Details

The following are examples of the information that you can get about the objects in your deployment.

- The application is *hello_world-app*.
- The application process is *Deploy*, which has one application process step called *put html file*.
- The application process step calls a component process called *Deploy*, which has two steps: *get html files* and *copy to apache*.

Example:

```
Job: 3_Deploy_hello_world-app_Default_20150304190119


Workspace File – Retrieve Artifact.d7fc4f54-c2a0-11e4-acf4-0800276bd168.log
Source: /home/eccloud/sample_dsl/hello_world
Artifact Name: hello_world.html
Artifact Version: 1
Destination directory: /opt/electriccloud/electriccommander/workspace/3_Deploy_hello_world-app_Default_20150304190119
Fetching artifact: done.
```

Example 2: Application Process Step Level Details

When you click the **View Details** button for **put html files**, you go to the Job Details page for this job step. You can click the Job link in the General Information to view information about the job in the previous example.

Job: 3_Deploy_hello_world-app_Default_20150304190119

Job Step Details – put html files



Completed with Success

Start Time: 2015-03-04 19:01:20 UTC
Elapsed Time: 00:00:01.024

General Information

Job: 3_Deploy_hello_world-app_Default_20150304190119
Procedure: External

General | Diagnostics | Properties | Notifiers

General

Step Id: d7b09fe2-c2a0-11e4-b341-0800276bd168
Step Name: put html files
Subproject: [Default](#)
Create Time: 2015-03-04 19:01:19 UTC
Elapsed Time: 00:00:01.024
End Time: 2015-03-04 19:01:21 UTC
Last Modified: 2015-03-04 19:01:21 UTC
Last Modified By: project: Default
Run Condition: 1
Exclusive Mode: none
Release Mode: none

Diagnostics

Exit Code: 0
Error Handling: Fail Procedure

Command

Exclusive Mode: none
Release Mode: none

Example 3: Component Process Step Details

When you click the **View Details** button for **get html files**, you go to the Job Details page for this job step. You can click the Job link in the General Information to view information about the job in the previous example.

Job: 3_Deploy_hello_world-app_Default_20150304190119

Job Step Details – get.html.files Access Control

Completed with Success
 Start Time: 2015-03-04 19:01:20 UTC
 Elapsed Time: 00:00:00.489

General Information
 Job: 3_Deploy_hello_world-app_Default_20150304190119
 Procedure: External

General | Diagnostics | Properties | Notifiers

General

Step Id: d7d3df43-c2a0-11e4-acf4-0800276bd168
 Step Name: get.html.files
 Subproject: /plugins/EC-FileSysRepo/project
 Create Time: 2015-03-04 19:01:19 UTC
 Elapsed Time: 00:00:00.489
 End Time: 2015-03-04 19:01:20 UTC
 Last Modified: 2015-03-04 19:01:20 UTC
 Last Modified By: project: Default
 Run Condition: 1
 Exclusive Mode: none
 Release Mode: none

Diagnostics

Exit Code: 0
 Error Handling: Fail Procedure

Command

Exclusive Mode: none
 Release Mode: none

Example 4: Change History of a Resource

When you click the **Track Changes** button for resource called *hello_world-web-server-resource*, you go to the Change History for Details page for the application process.

Change History for Deploy

2/12/15 - 3:30 PM **22** 3/20/15 - 1:45 PM

Between... and...

View All Changes	When	What	Name	By...	Change	Path
Objects	1 Mar 04, 2015 10:58 AM Pacif...	process	deploy	admin	* created	+/+
Acl (9)	2 Mar 04, 2015 10:58 AM Pacif...	property	ec_notifier...	admin	* created	+/+
Property Sheet (4)	3 Mar 04, 2015 10:58 AM Pacif...	property	ec_deploy	admin	* created	+/+
✓ Process (1)	4 Mar 04, 2015 10:58 AM Pacif...	processStep	put.html.files	admin	* created	+/+
✓ Property (4)	5 Mar 04, 2015 10:58 AM Pacif...	formalPara...	ec_hello_...	admin	* created	+/+
✓ Process Step (1)	6 Mar 04, 2015 10:58 AM Pacif...	formalPara...	ec_hello_...	admin	* created	+/+
✓ Formal Parameter (3)	7 Mar 04, 2015 10:58 AM Pacif...	formalPara...	ec_smartd...	admin	* created	+/+
Changes	8 Mar 04, 2015 10:58 AM Pacif...	property	ec_notifier...	admin	* created	+/+
✓ Created (22)	9 Mar 04, 2015 10:58 AM Pacif...	property	ec_deploy	admin	* created	+/+
Changed by...						
✓ Admin (22)						

Searching the Change History

Follow these steps to start a search in the Change History.

You can start a Change History search from most pages in the ElectricFlow UI.

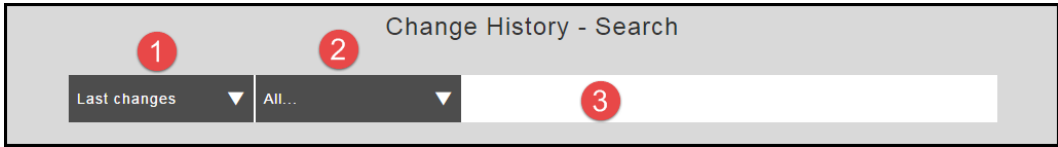
- 1. Click the **Search** button or click **Change History**.

Example:



The **Change History - Search** dialog box opens.

Example:

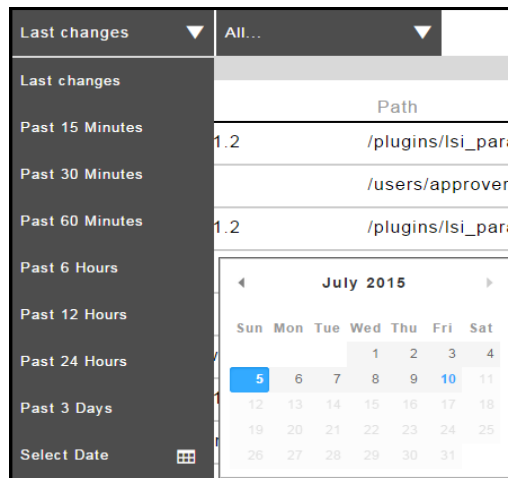


1	<p>Time range field.</p> <p>Click the down arrow to open the drop-down list of start times.</p> <p>The end time is the current time.</p>
2	<p>Objects field.</p> <p>Click the down arrow to open the drop-down list of objects to include in the search. You can select All or specific objects.</p> <p>By default, seven of the most commonly tracked objects are selected.</p>
3	<p>Search criteria.</p> <p>After you type, the system starts searching for objects based on the time range and objects that you selected.</p> <p>The search results are in the Change History.</p>

2. To select a time range for the search:
 - a. Click the down arrow in the **Time range** field to open the drop-down list.
 - b. Select a time range.
 - c. If you want to use a time increment longer than three days, do the following:
 - a. Click **Select Date**.

The Date Picker opens.

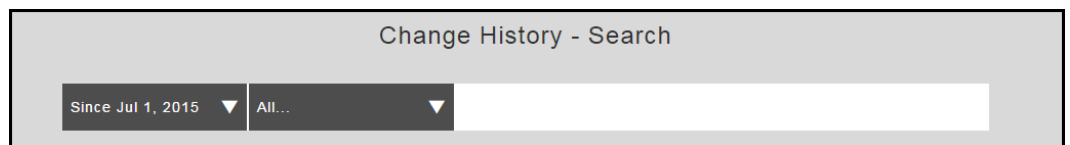
Example:



- b. Select a date.

The Date Picker closes and the date that you selected appears in the Time Increment field.

Example:



3. To select an objects for the search:
 - a. Click the down arrow in the **Object** field to open the drop-down list.
 - b. Select the objects for the search.
4. Enter the search criteria.

As you type, the system starts to search for objects that match your search criteria.

A list of objects matching your search criteria appears in the results section.

5. Select an object in the list.

Example:

Since Jul 1, 2015	All...	property	X
Access Control En...	ecscm-property-2.0.1.6...	/plugins/ecscm-property-2.0.1.65837	
	propertyviewer-1.0.1	/plugins/propertyviewer-1.0.1	
Access Control List	propertyviewer-1.0.1	/plugins/propertyviewer-1.0.1	
	ecscm-property-2.0.1.6...	/plugins/ecscm-property-2.0.1.65837	
Plugin	statetemplate_fullprope...	/server/ec_notifiertemplates/statetemplate_fullpropertypaths	
	propertyviewer-1.0.1	/plugins/propertyviewer-1.0.1	
Property	ecscm-property-2.0.1.6...	/plugins/ecscm-property-2.0.1.65837	
	artifactversionlocationp...	/projects/default/applications/heat clinic (killer app)/component...	
	artifactversionlocationp...	/projects/default/applications/jpetstore-aws-beanstalk/compone...	
	label	/server/ec_notifiertemplates/statetemplate_fullpropertypaths/la...	
	artifactversionlocationp...	/projects/default/applications/heat clinic store 1.1/components/...	
	artifactversionlocationp...	/projects/default/applications/heat clinic (killer app)/component...	
	artifactversionlocationp...	/projects/default/applications/heat clinic store 1.1/components/...	
	artifactversionlocationp...	/projects/default/applications/heat clinic store 1.1/components/...	
	series 1 property function	/projects/electric cloud/schedules/report recent job trend/series...	
	artifactversionlocationp...	/projects/default/applications/heat clinic (killer app)/component...	

The change history for the object that you selected appears.

Example:

Change History for propertyviewer-1.0.1						
<div> <div>▼ Last changes</div> <div> <div>7/10/15 - 1:02 PM</div> <div>Now - 7/10/15 - 6:18 PM</div> </div> </div>						
View All Changes	When	What	Name	By...	Change	Path
Objects						
✓ Acl (1)	1 Jul 10, 2015	1:02 PM Pacific...	acl	propertyviewer-1.0.1	admin	created
✓ Ace (3)	2 Jul 10, 2015	1:02 PM Pacific...	ace	propertyviewer-1.0.1	admin	created
Changes						
✓ Created (4)	3 Jul 10, 2015	1:02 PM Pacific...	ace	propertyviewer-1.0.1	admin	created
Changed by...						
✓ Admin (4)	4 Jul 10, 2015	1:02 PM Pacific...	ace	propertyviewer-1.0.1	admin	created

Modeling and Deploying Applications in Dynamic Environments

You can model an application, deploy it in a dynamically created environment, and view and troubleshoot the results. In this workflow, you can create dynamic environments with cloud resources that are spun up when you deploy the application. It does not describe how to create static environments with resources in your system or network that are configured before deploying the application. For information about this workflow, go to [Modeling and Deploying Applications in Static Environments](#) on page 87

- An environment with only provisioned cloud resources.
- An environment with only static resources added to an environment template.
- An environment with provisioned cloud resources and static resources

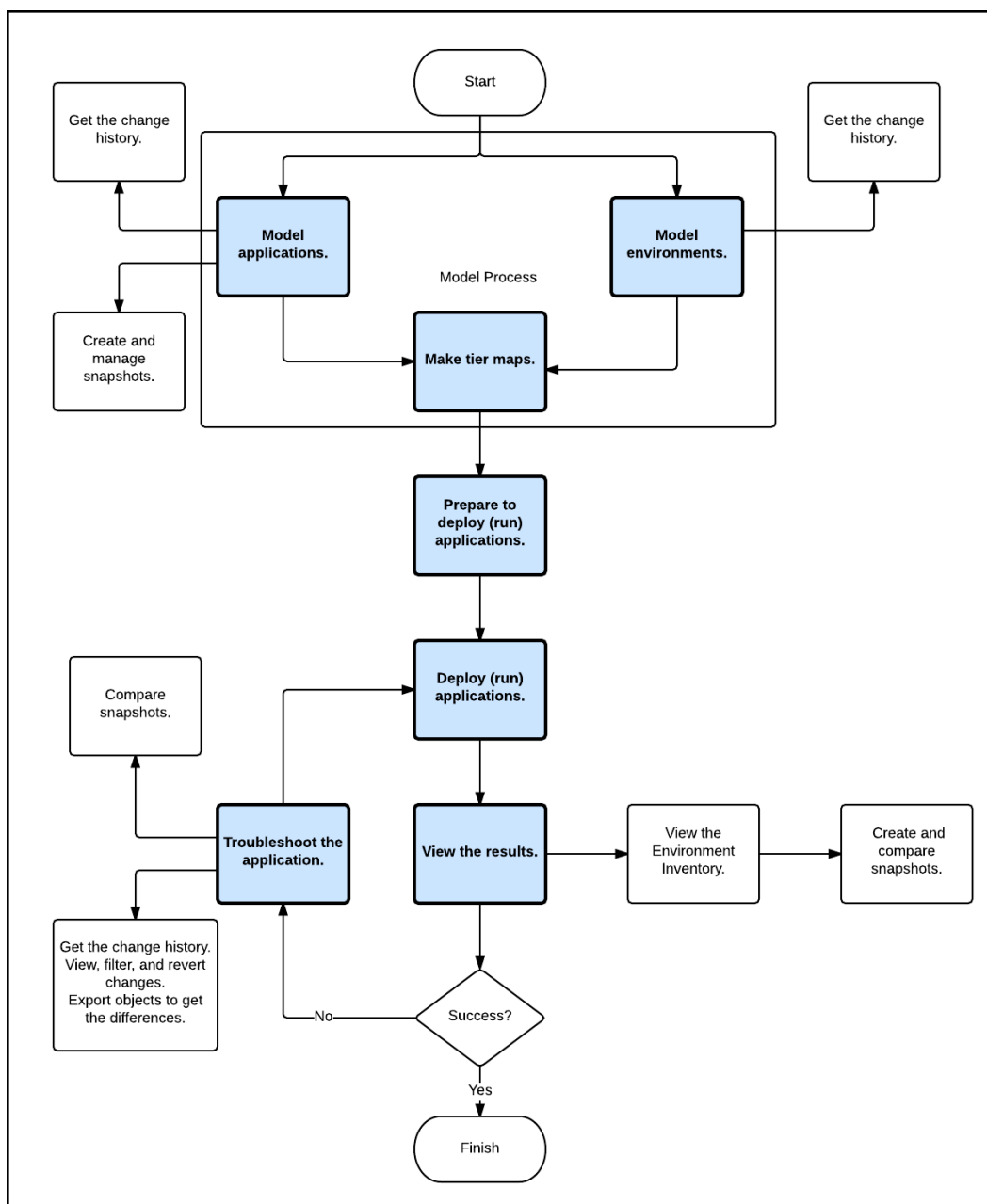
Using dynamic environments, teams that use dynamic cloud resources to deploy applications can do the following:

- Define and provision dynamic cloud resources when deploying applications.
- Configure the middleware of dynamic cloud resources on an on-demand basis.
- Re-use resource pools.
- Track how provisioned cloud resources are used.
- Provide ways to optimize how cloud resources are used.
- Provide status of the process that provisions cloud resources.
- Verify the credentials of cloud resources before provisioning them.

The following tasks describe how to model and deploy (run) applications at a high level.

1. [Logging in to ElectricFlow](#) on page 194
2. [Modeling Applications](#) on page 195
3. [Modeling Dynamic Environments](#) on page 247
4. [Deploying Applications With Provisioned Cloud Resources](#) on page 302
5. [Retiring Dynamic Environments](#) on page 311

For information about the UI, go to [ElectricFlow Buttons and Icons](#) on page 333.



More about application, deploy, and run:

As you use ElectricFlow, remember that these terms have different meanings within ElectricFlow *and* outside of ElectricFlow when you deploy your software or application:

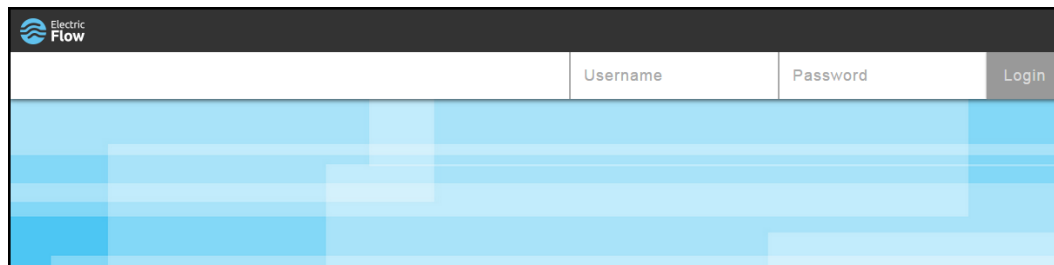
Term	Within ElectricFlow	Outside of ElectricFlow
Application	The application that you design and run (deploy) to produce your software for continuous delivery across different pipelines.	The software, system or application that you build, test, install, implement, release, and deploy using ElectricFlow. This is the end product of using ElectricFlow.
Deploy	Running the application that you designed in ElectricFlow. The end product is your software, system, or application. Deploy is a synonym of run in ElectricFlow.	All the processes or actions to develop and run your software in its environment, including building, testing, implementing, installing, configuring, making changes, and releasing.
Run	Running the application that you designed. The end product is your software, system, or application. <i>Run</i> is a synonym of <i>deploy</i> in ElectricFlow.	All the processes or actions to use software in its environment, including implementing, installing, configuring, debugging, troubleshooting, and releasing.

Logging in to ElectricFlow

1. Enter **http://<ElectricFlow-server>/flow** in a browser window, where <ElectricFlow-server> is the ElectricFlow server IP address or host name.

For example, when you go to <https://123.123.1.222/flow/>, the landing page opens.

Example:

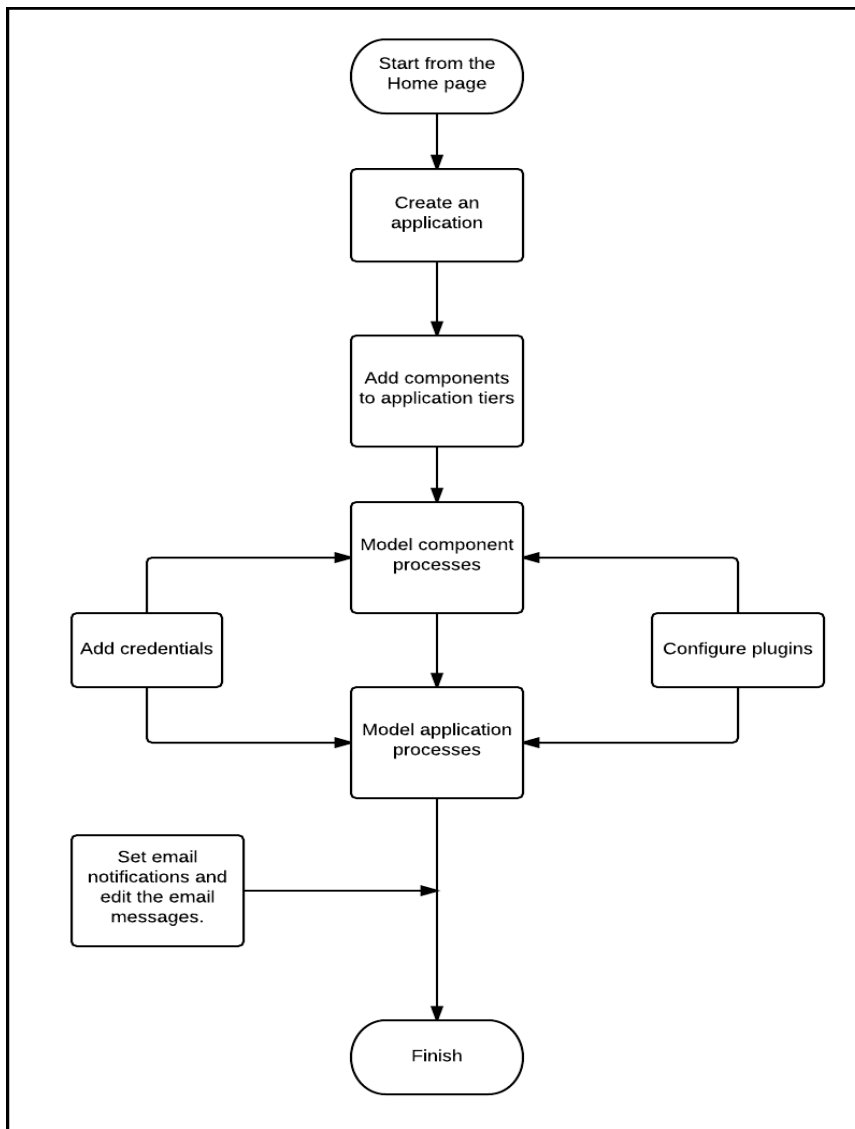


IMPORTANT: For a new installation, the default admin account user name is *admin* and the password is *changeme*. You should change the default admin password as soon as possible.

2. Enter a user name and password.
3. Click **Login**.

The ElectricFlow Home page opens.

Modeling Applications



1. Create an application and application tiers.
2. Add components to the application tiers.
3. Model component processes.
4. Model application processes.
5. (Optional) Set email notifications and edit the email messages.

Creating an Application and Application Tiers

Starting from the Home page:

1. Go to the Applications List by either
 - Clicking the **Applications** launch pad.
 - Clicking the **Menu** button > **Applications**.

The Applications List opens.

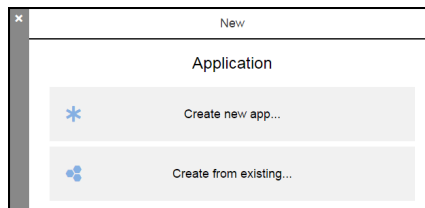
2. Click the **Add +** button in the upper right corner.

Example:



The New Application dialog box appears.

Example:



There are two ways to create an application:

- Click **Create new app** to create a new application.
 - Click **Create from existing** to create an application based on an existing application.
3. Create the application by going to appropriate next step.
 - To create a new application, go to [Creating a New Application](#) on page 197.
 - To create an application based on an existing application, go to the next step.

4. Click **Create from existing** to create an application based on an existing one.

The new application will have the same objects (components, artifacts, and application processes) as the existing application. However, it is not an exact copy of it because you need to configure new tier maps for it.

The **New Application from Existing** dialog box opens.

- a. Select an application.

The **New Application Name** dialog box opens with the name of the application you selected in the **Name** field.

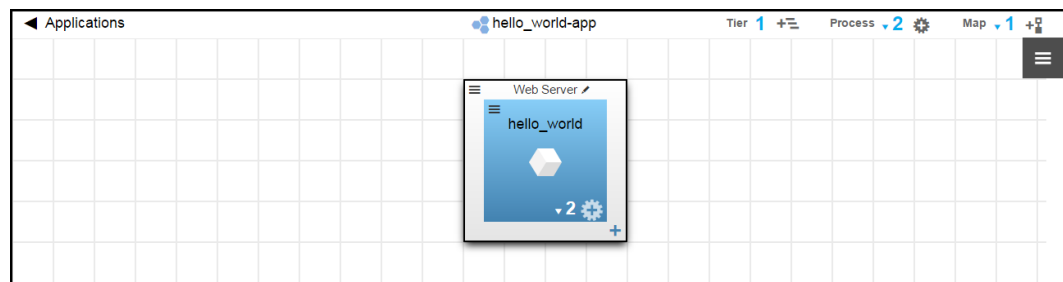
- b. Enter a name in the **Name** field.

It must not match the name of another application in the project.

- c. (Optional) Enter a description of the application in the **Description** field.

- d. Click **OK**.

If you are modeling an application based on an existing application, the Applications Visual Editor displays the same application tiers and components as the existing application with the name that you entered.



Creating a New Application

Starting in the Applications List:

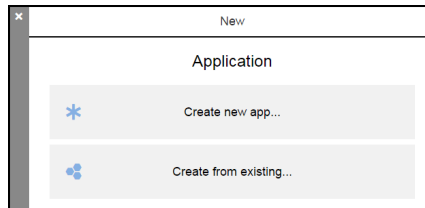
1. Click the **Add +** button in the upper right corner.

Example:



The New Application dialog box appears.

Example:



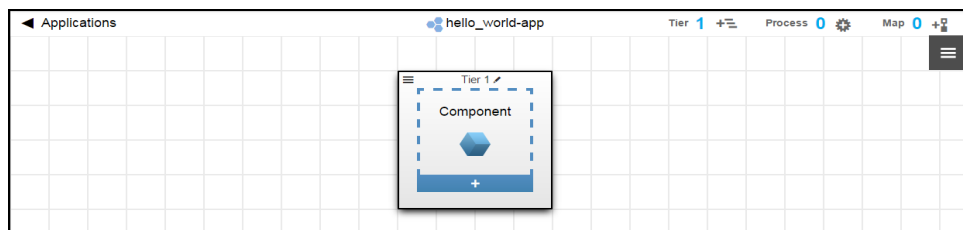
2. Click **Create new app** to create an application.

The **New Application Name** dialog box opens.

- a. Enter a name in the **Name** field.
- b. (Optional) Enter a description of the application in the **Description** field.
- c. Click **OK** to save the settings.

If you are modeling a new application, the Applications Visual Editor displays an application tier called Tier 1 with one component called Component.

Example:



3. Go to [Defining Components](#) on page 360 to set the component details.

Adding Components to the Application Tiers

Starting in the Applications Visual Editor:

1. Click the **Edit** button.

The **Application Tier Details** dialog box opens.

2. Change the name of the tier and click OK.
3. Click the **+** button in the component.

The **New Component** dialog box opens.

4. Enter a name in the **Name** field.
5. (Optional) Enter a description of the component in the **Description** field.
6. Click **Next** to save the settings.

The Component Details dialog box opens.

- Click the **Current Location** field.

A list of available artifact locations appears.

- Select a location and click **Browse**.

The information needed to define the artifact appears below.

- Enter the appropriate information in the fields.

Example:

Component Details

hello_world Description

EC-FileSysRepo Browse

Source: /home/eccloud/sample_dsl/hello_ Required

Artifact: hello_world.html Required

Version: ☒ Latest

☒ Exact 1

Retrieve to Directory:

Cancel OK

1	Content Location of the component
2	Component details that vary depending on the Content Location

10. Click **OK**.

The Applications Visual Editor now shows an application tier called Web Server with a component called hello_world.

Example:



11. To add a component to the same tier, click the **+** button in the lower right corner of the tier.

A new undefined component appears in the tier.

12. Configure this component the same way you did the first one.

Adding a Component

To add component to the same application tier:

1. Click the **+** button in the lower right corner of the tier.

A new undefined component appears in the tier.

2. Click the **+** button in the new component to set the component details, following the steps previously described.

Adding an Application Tier

To add an application tier, click the **Add tier** button.

Modeling Component Processes

Starting in the Applications Visual Editor:

1. Click the **Add Process** button in a component to a component process to it.

Example:



The **New Component Process Details** dialog box opens.

2. Enter information about the component process in the Component Process Details dialog box.

Example:

The screenshot shows a 'New Component Process Details' dialog box. It features a title bar labeled 'New' and a main heading 'Component Process Details'. The form includes a 'Name' text field, a 'Description' text area, a 'Deploy' dropdown menu, a 'Credential' field with the value '0' and an 'Optional' link, a 'Workspace' dropdown menu with 'default' and an 'Optional' link, and a 'Time limit' field with '0' and a 'Seconds' dropdown menu with an 'Optional' link. The bottom of the dialog contains 'Cancel' and 'OK' buttons.

Field	Description and How to Set It
Name	Name of the process step
Description	Description of the process step
Process Type	<p>Type of process. The default is Deploy.</p> <p>To set the process type:</p> <ol style="list-style-type: none"> 1. Click the Type field to select the process type. 2. Select one of these options: <ul style="list-style-type: none"> Deploy—Enables Inventory Tracking. The ElectricFlow server tracks the artifacts deployed to environments. Undeploy—The next time that the process is run, the ElectricFlowserver removes information about the artifacts deployed to environments. Other—Disables Inventory Tracking.

Field	Description and How to Set It
Credential	<p>An object consisting of a user name and password that ElectricFlow uses to run a process step.</p> <p>The dialog box displays the number of credentials for the process step, which are the same credentials that you use with procedures, steps, and schedules in the automation platform.</p> <p>You can only impersonate one credential. See Adding Credentials on page 232 to set the process type.</p>
Workspace	<p>Area in the disk space where the files and results of the job step are stored.</p> <p>To set the workspace, click the Workspace field to open a drop-down list of workspaces in the ElectricFlow platform and select a workspace.</p> <p>For more information about workspaces, go to the "Workspaces and Disk Management" topic. To set the workspace, click Workspace to open a drop-down list of workspaces in the ElectricFlow platform. Select a workspace, and click OK.</p>
Time limit	<p>Maximum length of time that the step is allowed to run. After the time specified, the step is aborted,</p> <p>To set the time limit, enter the time and select the unit of time: seconds, minutes, or hours.</p> <p>For information about time limits for procedure job steps in the ElectricFlow platform, go to the "Job Step Details" topic.</p>

Example:

New

Component Process Details

Deploy

Description

Deploy ▼

Credential

0

Optional >

Workspace

Optional

Time limit

0

Seconds ▼

Optional

Cancel

OK

- (Optional) To add credentials, do the following:

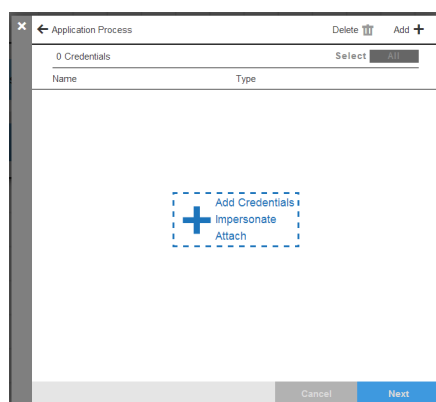
IMPORTANT:

When you impersonate a credential, make sure that the impersonated user has the absolute path to the bin directories in the \$PATH environment.

If you define a process step with a command, you must enter the absolute path in the **Post Processor** and **Shell** fields in the Define Step dialog box.

- Click in the **Add Credentials** field.

Example:



- To impersonate one credential, select **Impersonate** in the **Type** field.
- Click the **Select Credential** field to open a drop-down list of credentials for the process step.
- Select a credential.
- Click **OK**.

The **Credentials** dialog box now shows the one credential for impersonation.

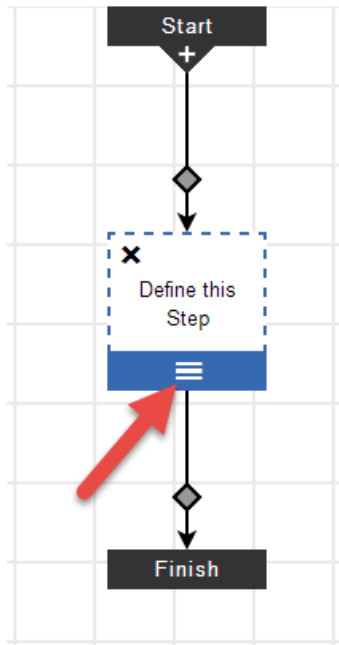
- To attach one or more credential to the process step, select **Attach** in the **Type** field.
- Click the **Select Credential** field to open a drop-down list of credentials for the process step.
- Select a credential.
- Click **OK**.

The **Credentials** dialog box now shows the attached credentials.

- Click **OK**.

The Component Process Visual Editor opens.

5. In the new process step, click the button below "Define this Step" to define it.

Example:

The Component Process Step dialog box opens.

6. Enter information about the step in the dialog box.

Example:

New Step

Component Process Step

Step name: Required

Description:

Credential: >

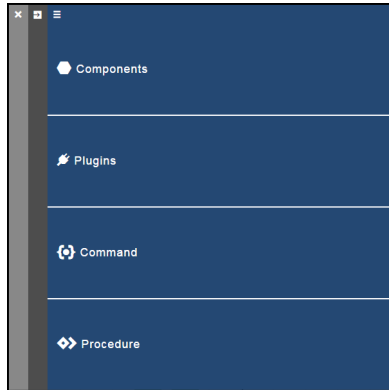
Workspace: ▼

Time limit: Seconds ▼

7. Click **Next**.

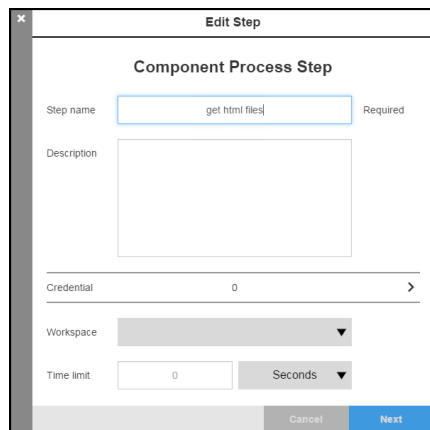
The Process Step dialog box opens.

Example:



8. To define the step, enter information in the dialog boxes that follow.

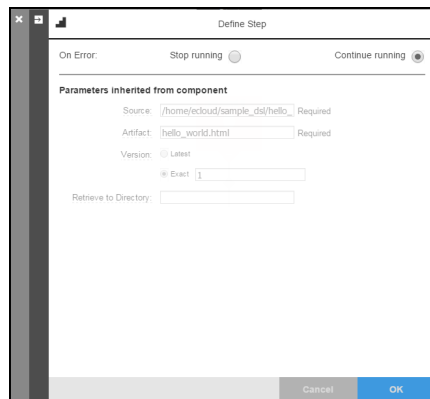
Examples:



The 'Edit Step' dialog box is titled 'Component Process Step'. It contains the following fields:

- Step name:** A text box containing 'get.html.files' with a 'Required' label to its right.
- Description:** A large empty text area.
- Credential:** A dropdown menu showing '0' with a right-pointing arrow.
- Workspace:** A dropdown menu.
- Time limit:** A text box containing '0' and a 'Seconds' dropdown menu.

At the bottom are 'Cancel' and 'Next' buttons.



The 'Define Step' dialog box is titled 'Define Step'. It contains the following fields:

- On Error:** Two radio buttons: 'Stop running' (unselected) and 'Continue running' (selected).
- Parameters inherited from component:**
 - Source:** A text box containing '/home/ec2cloud/sample_ds/hello_' with a 'Required' label.
 - Artifact:** A text box containing 'hello_world.html' with a 'Required' label.
 - Version:** Two radio buttons: 'Latest' (selected) and 'Exact' (unselected).
 - Retrieve to Directory:** An empty text box.

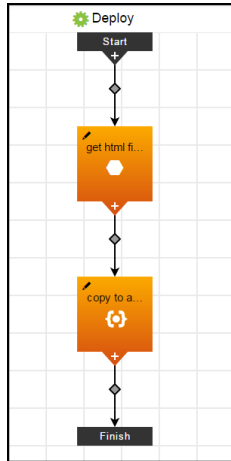
At the bottom are 'Cancel' and 'OK' buttons.

When you are done, the defined step now appears in the process in the Component Process Visual Editor.

9. Define more steps in the process.

You can also drag and drop a step into the process.

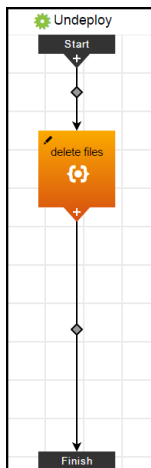
Example:



Modeling More Component Processes

Repeat the steps in the previous section to add additional component processes.

Example:



Modeling Application Processes

Starting in the Applications Visual Editor:

1. Click the **Add process** button in the upper right corner to add an application process.

Example:



The **New Application Process Details** dialog box opens.

2. Enter information in the New Application Process Details dialog box, and click **OK**.

Example:

New

Application Process Details

Name

Description

Credential

0

Optional >

Workspace

default

Optional

Time limit

0

Seconds

Optional

Cancel

OK

Field	Description and How to Set It
Name	Name of the process step
Description	Description of the process step
Credential	<p>An object consisting of a user name and password that ElectricFlow uses to run a process step.</p> <p>The dialog box displays the number of credentials for the process step, which are the same credentials that you use with procedures, steps, and schedules in the automation platform.</p> <p>You can only impersonate one credential.</p>

Field	Description and How to Set It
Workspace	<p>Area in the disk space where the files and results of the job step are stored.</p> <p>To set the workspace, click the Workspace field to open a drop-down list of workspaces in the ElectricFlow platform and select a workspace.</p> <p>For more information about workspaces, go to the "Workspaces and Disk Management" topic.</p> <p>To set the workspace, click Workspace to open a drop-down list of workspaces in ElectricFlow, select a workspace, and click OK.</p>
Time limit	<p>Maximum length of time that the step is allowed to run. After the time specified, the step is aborted,</p> <p>To set the time limit, enter the time and select the unit of time: seconds, minutes, or hours.</p> <p>For information about time limits for procedure job steps in ElectricFlow, go to the "Job Step Details" topic.</p>

- (Optional) To add credentials, do the following:

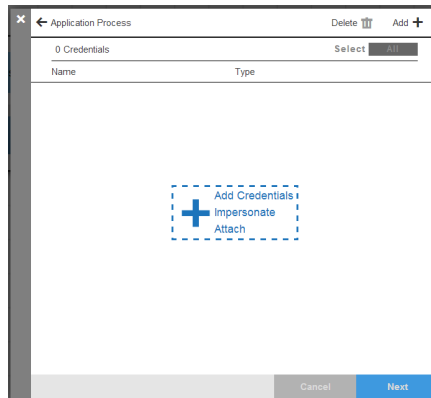
Note:

When you impersonate a credential, make sure that the impersonated user has the absolute path to the bin directories in the \$PATH environment.

If you define a process step with a command, you must enter the absolute path in the **Post Processor** and **Shell** fields in the Define Step dialog box.

- Click in the **Add Credentials** field.

Example:



- To impersonate one credential, select **Impersonate** in the **Type** field.
- Click the **Select Credential** field to open a drop-down list of credentials for the process step.
- Select a credential.
- Click **OK**.

The **Credentials** dialog box now shows the one credential for impersonation.

- To attach one or more credential to the process step, select **Attach** in the **Type** field.
- Click the **Select Credential** field to open a drop-down list of credentials for the process step.
- Select a credential.
- Click **OK**.

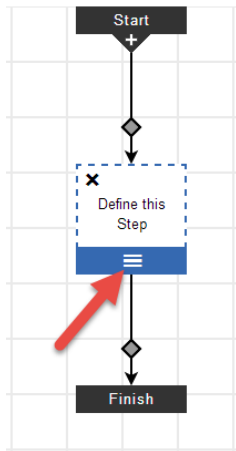
The **Credentials** dialog box now shows the attached credentials.

- Click **OK**.

The Applications Process Visual Editor opens.

5. In the new process step, click the button below "Define this Step" to define it.

Example:



The Application Process Step dialog box opens.

6. Enter information about the step in the dialog box.

Example

New

Application Process Details

Deploy

Description

Credential

0

Optional >

Workspace

Optional

Time limit

0

Seconds

Optional

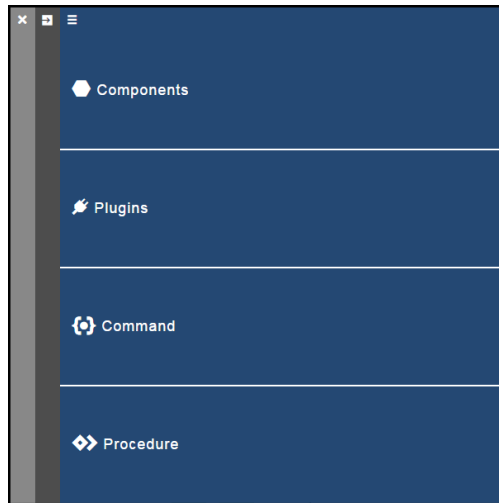
Cancel

OK

7. Click **Next**.

The Process Step dialog box opens.

Example:



8. To define the step, enter information in the dialog boxes that follow.

Example:

The 'Edit Step' dialog box is titled 'Edit Step' and contains the following fields:

- Step name:** A text box containing 'put.html.files', marked as 'Required'.
- Description:** A large empty text area.
- Tier:** A dropdown menu showing 'Web Server', marked as 'Required'.
- Credential:** A text box containing '0' with a right-pointing arrow.
- Workspace:** A dropdown menu.
- Time limit:** A text box containing '0' and a dropdown menu showing 'Seconds'.

At the bottom are 'Cancel' and 'Next' buttons.

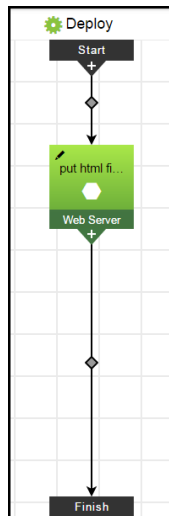
The 'Define Step' dialog box is titled 'Define Step' and contains the following elements:

- On Error:** Two radio buttons: 'Stop running' (unselected) and 'Continue running' (selected).
- Step Name:** A text box containing 'hello_world / Deploy'.

At the bottom are 'Cancel' and 'OK' buttons.

9. When you are done, the defined step now appears in the process in the Applications Process Visual Editor.

Example:



10. Define more steps in the process.

You can also drag and drop a step into the process.

Modeling More Application Processes

Repeat the steps in the previous section to add additional application processes.

Using the Drag and Drop Method to Add Process Steps

How to get to the Application Process Visual Editor:

- In an existing application process:

From the Applications Visual Editor, click the down arrow next to the number-of-application-processes button and select an application.

The Application Process Visual Editor for that application process appears.

- In a new application process:

From the Applications Visual Editor, click the **Add process** button, set the parameters in the **Application Process Details** dialog box, and click **OK**.

The Application Process Visual Editor for the application appears.

How to get to the Component Process Visual Editor:

- In an existing component process:

From the Applications Visual Editor, click the down arrow next to the number-of-component-processes button in a component, and select a component process in the drop-down list.

The Component Process Visual Editor for that component process appears.

- In a new component process:

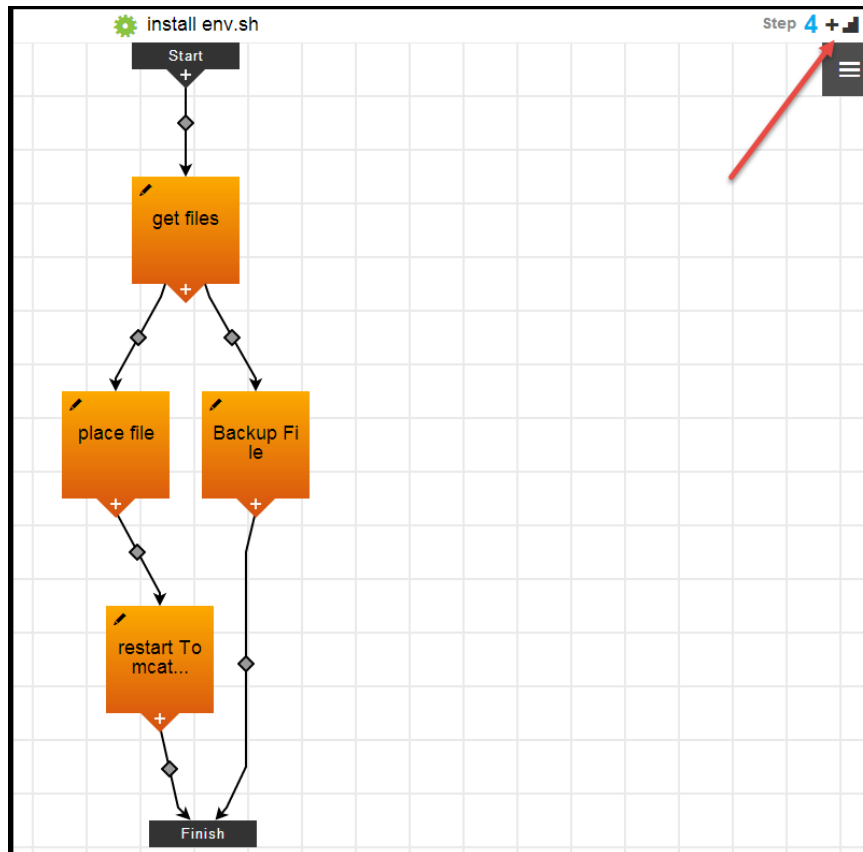
From the Applications Visual Editor, click the **Add process** button in a component, set the parameters in the **Component Process Details** dialog box, and click **OK**.

The Component Process Visual Editor for the component process appears.

To drag and drop a new step in a component or application process:

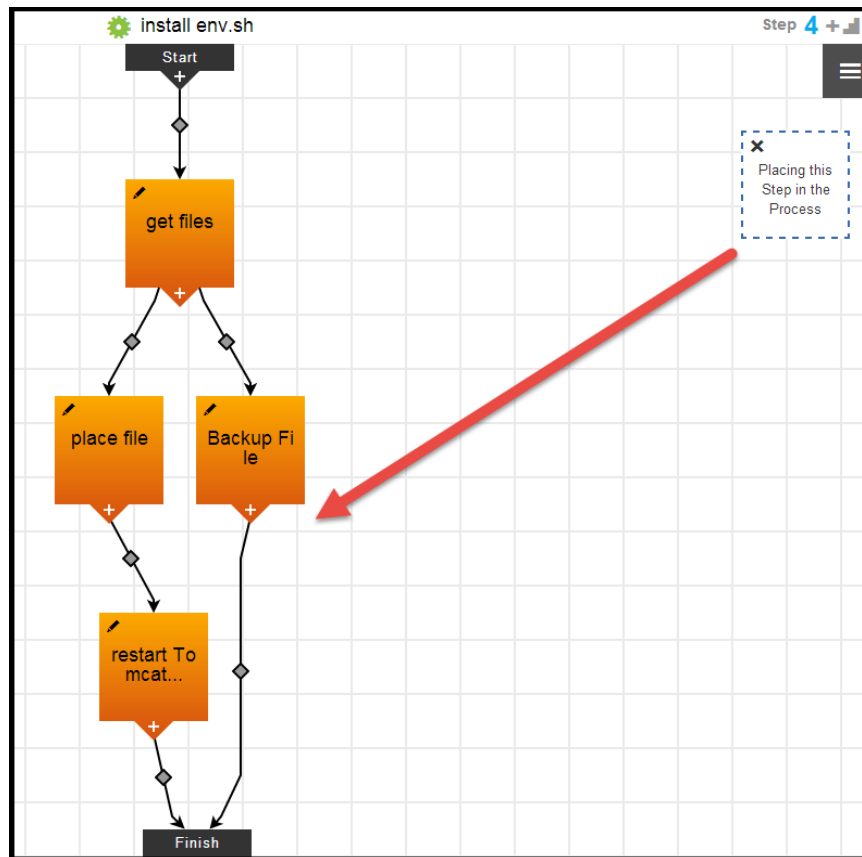
1. Click the **Add step** button in the upper right corner of the Component Process Visual Editor or Application Process Visual Editor.

.A new undefined step appears.



2. Select the new step.

3. Drag the step to where you want to add it in the process.



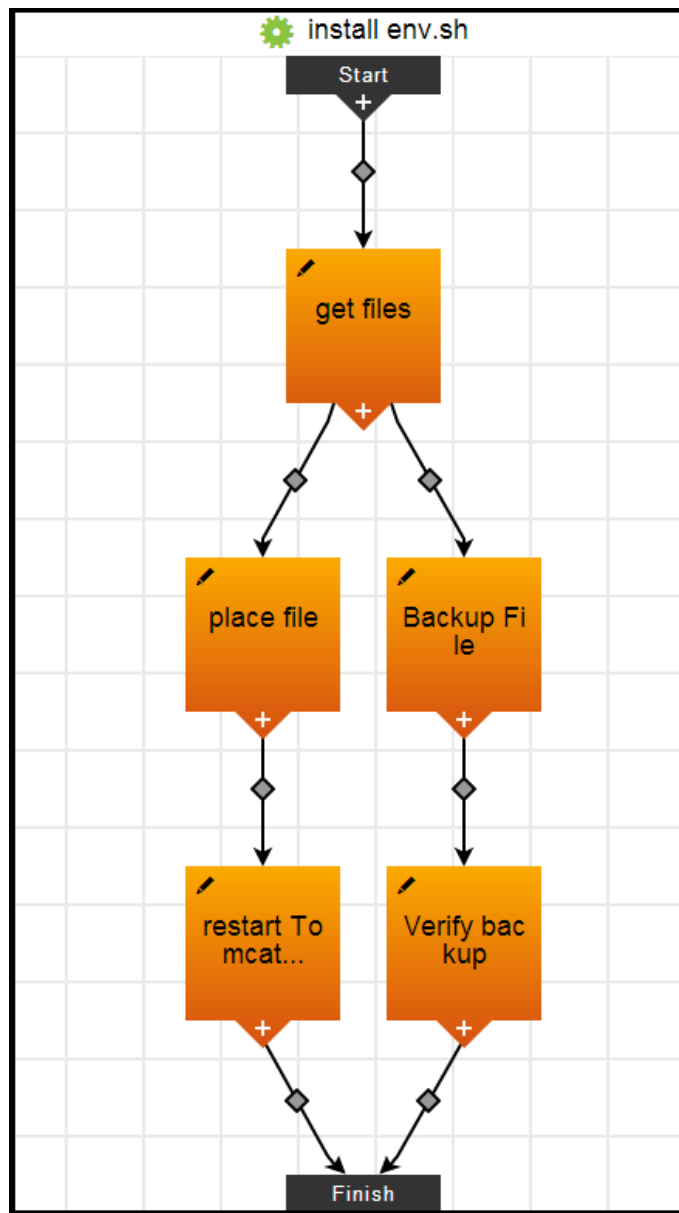
When you are near where you want to add the step in the process, notice that the icon changes shape and the text in it changes to "Dropping this Step in the Process."

4. Drop the step in the process.

The **Component Process Step** dialog box appears.

5. Enter information about the step.

The new step is in the process.

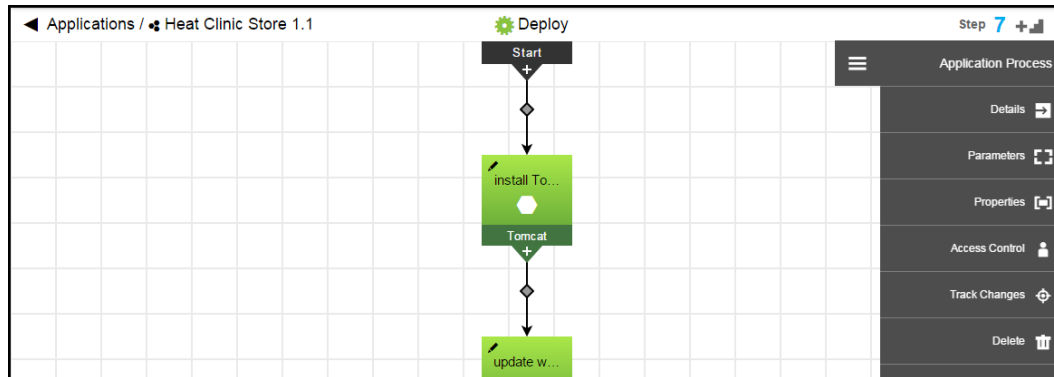


Setting Parameters for Application Processes

Starting in the Application Process Visual Editor:

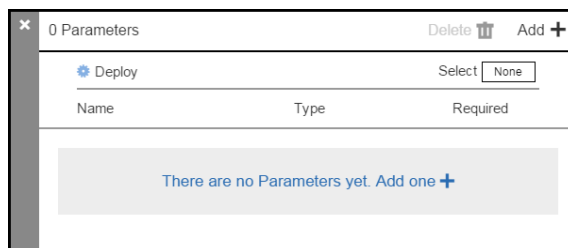
1. Click the **Menu** button.
2. Click **Parameters**.

Example:



The **Parameters** dialog box opens.

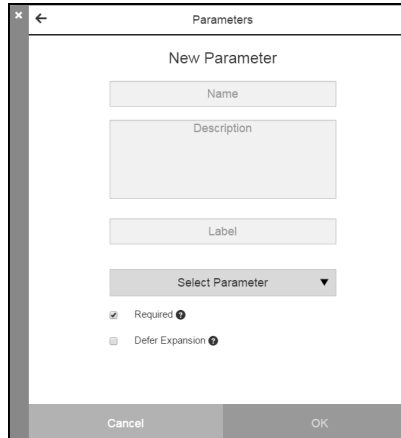
Example:



3. Click **There are no Parameters yet. Add one +**.

The **New Parameter** dialog box opens.

Example:



4. Enter the following information:

- **Parameter Name**—Name of the parameter.
- **Description**—This is optional.
- **Parameter Label**—This is optional.
- **Select Parameter**—Parameter type
- If the parameter is required, select the **Required** check box.
- If the parameter value contains \$ [] and you want ElectricFlow to interpret it literally (not as a parameter reference), select the **Defer Expansion** check box.

Depending on the parameter type that you select, other fields appear in the **New Parameters** dialog box.

5. Enter information in the fields in Step 4.

If you select **Text Entry** as the parameter type, the **Default Value** field appears.

- a. (Optional) Enter a value in the **Default Value** field.

You do not have to enter a value in this field.

Example:

The screenshot shows a 'Parameters' dialog box with a 'New Parameter' section. It contains the following fields and options:

- User Name**: A text input field.
- Text entry example**: A larger text input field.
- Label**: A text input field with a blue border.
- Cancel**: A small blue button next to the Label field.
- Text Entry**: A dropdown menu with a downward arrow.
- Default Value**: A text input field.
- Required**: A checked checkbox with a help icon.
- Defer Expansion**: An unchecked checkbox with a help icon.
- Cancel**: A grey button at the bottom left.
- OK**: A blue button at the bottom right.

- b. Click **OK**.

The **Parameters** dialog box opens and shows the new parameter in the list.

Example:

The screenshot shows the 'Parameters' dialog box with a list of parameters. The list has columns for Name, Type, and Required. The first parameter is 'User Name' of type 'Text Entry' and is required.

1 Parameters			Delete	Add
Deploy			Select	All
	Name	Type	Required	
1	User Name	Text Entry	✓	>

6. Click **Add +** to add a parameter.

7. Enter information in the fields in Step 4.

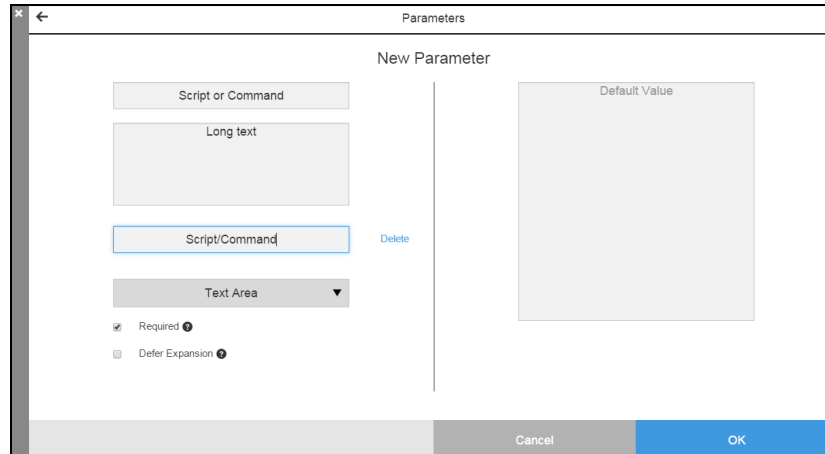
If you select **Text Area** as the parameter type, the **Default Value** field appears.

- a. (Optional) Enter a value in the **Default Value** field.

You do not have to enter a value in this field.

- b. Enter the Click **OK**.

Example:



The screenshot shows the 'Parameters' dialog box with a 'New Parameter' section. On the left, there are input fields for 'Script or Command', 'Long text', and 'Script/Command', followed by a 'Text Area' dropdown menu. Below these are checkboxes for 'Required' (checked) and 'Defer Expansion' (unchecked). On the right, there is a large 'Default Value' text area. A 'Delete' button is located between the input fields and the 'Default Value' area. At the bottom, there are 'Cancel' and 'OK' buttons.

The **Parameters** dialog box opens and shows the new parameter in the list.

8. Enter information in the fields in Step 4. If you select **Dropdown Menu** as the parameter type, the **Default Value** field and ways to add the menu options appear.
 - a. (Optional) Enter a value in the **Default Value** field.

You do not have to enter a value in this field.
 - b. On the right side of the dialog box, enter the menu options.
 - c. Click **OK**.

The **Parameters** dialog box opens and shows the new parameter in the list.

Example:

The screenshot shows the 'Parameters' dialog box with the 'New Parameter' form. The form is divided into two main sections. The left section contains fields for 'Stage', 'Stage of the deployment process', 'Label', 'Dropdown Menu' (set to 'Dropdown Menu'), and 'Default Value'. Below these fields are checkboxes for 'Required' (checked) and 'Defer Expansion' (unchecked). The right section shows three radio buttons for 'Enter options' (selected), 'Load options from list', and 'Load options from property sheet'. Below these is an 'Add option +' button. A list of three options is shown: 1. 'Development' / 'Dev', 2. 'Quality' / 'QA', and 3. 'Release' / 'Prod'. The 'Prod' option is highlighted. At the bottom are 'Cancel' and 'OK' buttons.

The **Parameters** dialog box opens and shows the new parameter in the list.

9. Enter information in the fields in Step 4.

If you select **Radio Selector** as the parameter type, the **Default Value** field and ways to add the menu options appear.

- (Optional) Enter a value in the **Default Value** field.
You do not have to enter a value in this field.
- On the right side of the dialog box, enter the menu options.
- Click **OK**.

Example:

The screenshot shows the 'Parameters' dialog box with the 'New Parameter' tab active. On the left, a list of parameters includes 'Ranking', 'Ranking in the queue', 'Priority', 'Radio Selector' (selected), and 'Normal'. A 'Delete' link is next to 'Radio Selector'. Below the list are checkboxes for 'Required' (checked) and 'Defer Expansion' (unchecked). On the right, the 'Enter options' radio button is selected. Below it are three rows of menu options: 1. 'High' (selected), 2. 'Normal', and 3. 'Low' (highlighted). Each row has a delete 'x' button. An 'Add option +' button is at the top right of the options section. At the bottom are 'Cancel' and 'OK' buttons.

The **Parameters** dialog box opens and shows the new parameter in the list.

10. Enter information in the fields in Step 4.

If you select **Checkbox** as the parameter type, the **Default Value** field and values for the check box appear.

- (Optional) Enter a value in the **Default Value** field.
You do not have to enter a value in this field.
- On the right side of the dialog box, enter the values.
- Click **OK**.

Example:

The screenshot shows a 'Parameters' dialog box with a 'New Parameter' section. On the left, there is a list of parameters: 'Ranking', 'Ranking in the queue', 'Priority', 'Radio Selector', and 'Normal'. The 'Radio Selector' is currently selected. Below the list, there are checkboxes for 'Required' (checked) and 'Defer Expansion' (unchecked). On the right, there are three radio buttons for 'Enter options', 'Load options from list', and 'Load options from property sheet'. Below these, there is a list of three options: 1. 'High', 2. 'Normal', and 3. 'Low'. Each option has a corresponding input field and a delete button (X). The 'Low' option is currently selected. At the bottom, there are 'Cancel' and 'OK' buttons.

The **Parameters** dialog box opens and shows the new parameter in the list.

11. Enter information in the fields in Step 4. If you select **Credentials** as the parameter type, the **Default Value** field appears.
- a. (Optional) Enter a value in the **Default Value** field.
- You do not have to enter a value in this field.
- b. Click **OK**.

Example:

Parameters

New Parameter

Identity

Description

User Name and Password

Delete

Credentials

Default Value

☒ Required

☐ Defer Expansion

Cancel

OK

The **Parameters** dialog box opens and shows the new parameter in the list.

Example:

6 Parameters

DeleteAdd

Deploy

SelectAll

Name	Type	Required	
1. User Name	Text Entry	✓	>
2. Script/Command	Text Area		>
3. Stage	Dropdown menu	✓	>
4. Priority	Radio Selector	✓	>
5. QA Verification Required	Checkbox		>
6. User Name and Password	Credential	✓	>

Setting and Modifying the Parameter Label

Starting in the **New Parameter** dialog box:

Example:

1. Add a label in the **Label** .

Example:

This is an optional task when you set parameters.

If you enter a label, it appears in the UI form when you deploy the application. If you do not enter a label, the parameter name appears in the UI form when you deploy the application.

2. Click **OK**.

3. To modify the label:
 - a. Open the **New Parameter** dialog box.
 - b. Clear the **Label** field.
 - c. Enter a new label.

Example:

The screenshot shows the 'New Parameter' dialog box. On the left side, there are several input fields: 'QA Verification Required', 'Description', 'Verify by QA' (highlighted with a red box), 'Checkbox' (with a dropdown arrow), and 'Default Value'. To the right of the 'Verify by QA' field is a 'Delete' button. On the right side of the dialog, there are two input fields: 'Value when unchecked' (containing 'No') and 'Value when checked' (containing 'Yes'). Below these is an 'Initially checked' checkbox, which is currently unchecked. At the bottom left, there are two checkboxes: 'Required' and 'Defer Expansion', both with question mark icons.

- d. Click **OK** to save the change.

4. To delete the label:
- a. Click **Delete** next to the **Label** field.

Example:

The screenshot shows a 'Parameters' window with a 'New Parameter' section. On the left, there are input fields for 'QA Verification Required', 'Description', 'Verify by QA', 'Checkbox', and 'Default Value'. Below these are checkboxes for 'Required' and 'Defer Expansion'. On the right, there are input fields for 'Value when unchecked' (containing 'No') and 'Value when checked' (containing 'Yes'), along with an 'Initially checked' checkbox. A blue 'Delete' button is highlighted with a red rectangle next to the 'Verify by QA' field. At the bottom are 'Cancel' and 'OK' buttons.

The text in the Label field changes.

Example:

This screenshot shows the same 'New Parameter' dialog box, but with a confirmation prompt. A blue button labeled 'Yes, delete this Label' is highlighted with a red rectangle, next to a smaller blue 'Cancel' button. The rest of the dialog box, including the input fields and the bottom 'Cancel'/'OK' buttons, remains the same as in the previous screenshot.

b. Click in the **Label** field.

The label disappears.

Example:

Parameters

New Parameter

QA Verification Required

Description

Label

Checkbox

Default Value

Required

Defer Expansion

Value when unchecked

No

Value when checked

Yes

Initially checked

Cancel

OK

c. Click **OK**.

The **Parameters** dialog box opens. The parameter name now appears in the name column.

Example:

6 Parameters

Delete

Add

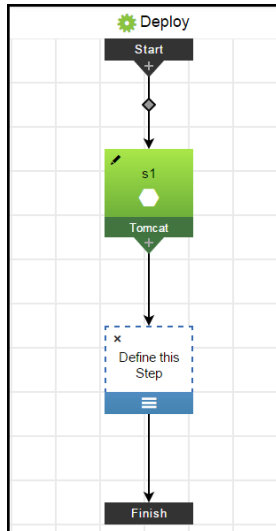
Deploy		Select All	
Name	Type	Required	
1. User Name	Text Entry	✓	>
2. Script/Command	Text Area		>
3. Stage	Dropdown menu	✓	>
4. Priority	Radio Selector	✓	>
5. QA Verification Required	Checkbox		>
6. User Name and Password	Credential	✓	>

Looking Up Parameters

To apply parameters to an application or component process step, starting in the Application Process Visual Editor:

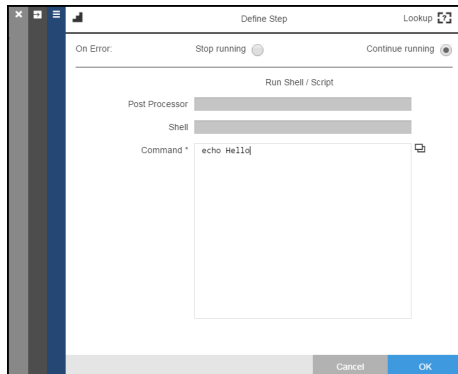
1. Add a new step to the process.

Example:



2. Define the process step in the **Define Step** dialog box.

Example:

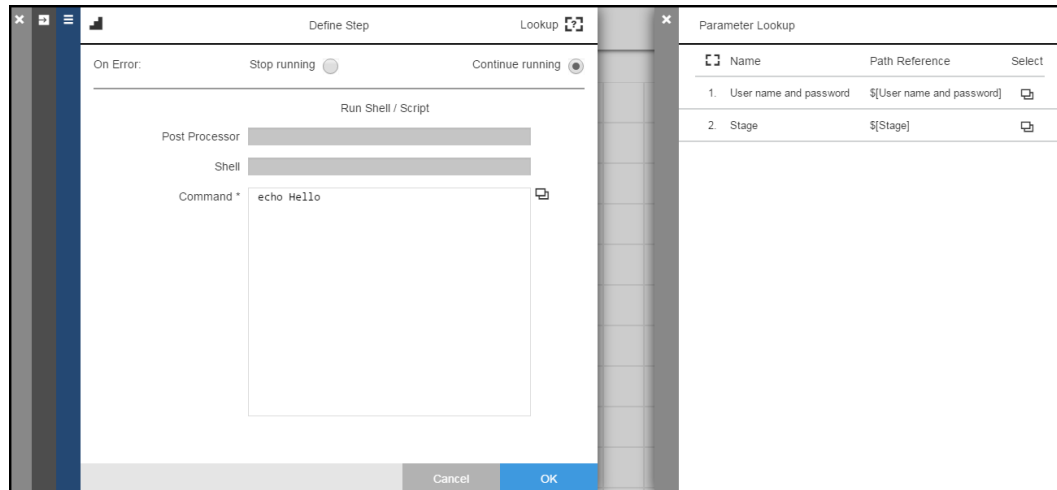


- When you define the process step with a plugin (**Plugins**), command or script (**Command**), or project (**Procedure**), click the **Lookup** button to open the **Parameter Lookup** dialog box.

Example:



The **Parameter Lookup** dialog box opens.



- Choose a parameter and click the **Copy** button to copy the path reference.

A message appears in the row : *<Parameter Name>* has been copied.

The **Define Step** dialog box now has a **Parameter** field.

- Click in the **Parameter** field and paste the path reference that you copied in it.
- Repeat the previous two steps to apply another parameter to the process step.
- Click **OK**.

Adding Credentials

How to get here: From the **Component Process Step** or the **Application Process Step** dialog box, click **>**. The **Credentials** dialog box opens.

You can attach one or more credentials to component process steps and application process steps.

You attach only one credential for impersonation on the following:

- Component process
- Component process step

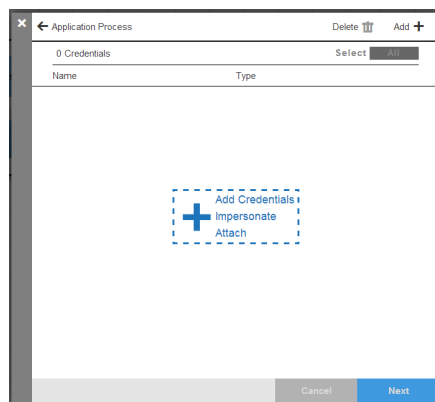
- Application process
- Application process step

IMPORTANT:

When you impersonate a credential, make sure that the impersonated user has the absolute path to the bin directories in the \$PATH environment.

If you define a process step with a command, you must enter the absolute path in the **Post Processor** and **Shell** fields in the Define Step dialog box.

1. Click in the **Add Credentials** field.

Example:

2. To impersonate one credential, select **Impersonate** in the **Type** field.
3. Click the **Select Credential** field to open a drop-down list of credentials for the process step.
4. Select a credential.
5. Click **OK**.

The **Credentials** dialog box now shows the one credential for impersonation.

6. To attach one or more credential to the process step, select **Attach** in the **Type** field.
7. Click the **Select Credential** field to open a drop-down list of credentials for the process step.
8. Select a credential.
9. Click **OK**.

The **Credentials** dialog box now shows the attached credentials.

Using Plugins

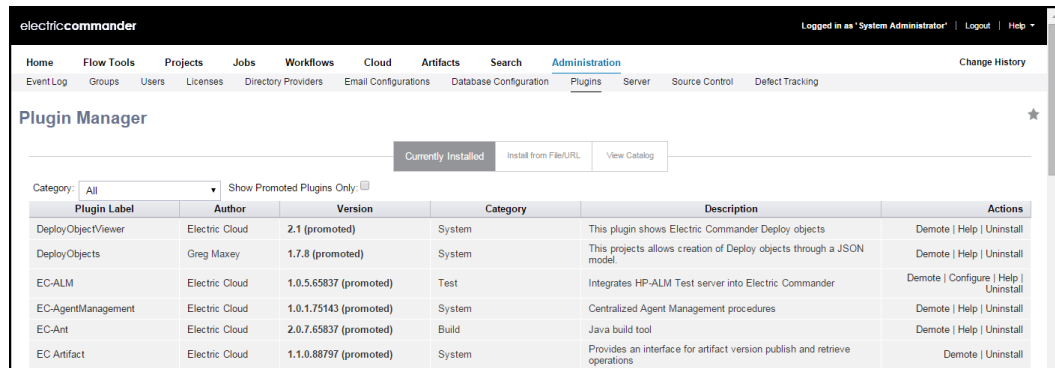
When you want to use a plugin to define your application or component process step, all of the supported plugins appear in the Plugin Manager in the **Currently Installed** tab in the automation platform. However, you may want to see only the list of plugins that apply to your group or organization, such as Apache Subversion

(SVN) and Git plugins. You can remove any plugins that you are not using from the **Currently Installed** tab. You can re-install them later if you need them.

To remove plugins from the **Currently Installed** tab:

1. In the Home page, click the **Main menu** button and then click **Admin > Plugins**.

The Plugin Manager opens.



2. In the **Currently Installed** tab, select a plugin in the list.
3. In the Actions column for the selected plugin, click **Demote**.

The page refreshes.

The plugin is now inactive but is still in the list. If you want to use this plugin, click **Promote** to make it active.

4. If you want to remove the plugin from this list, click **Uninstall** to remove it from your system.

For more details, go to the automation platform Help > **Web Interface Help > Plugin Manager**. This page describes what you can do in the Plugin Manager, including how to install a new version of a plugin or add a new plugin.

You can see all of the plugins available from Electric Cloud in the **View Catalog** tab on the **Administration > Plugins** page. To show a list of plugins that you can install directly from Electric Cloud, do the following:

1. Click **Install** in the **Action** column.
2. Go to the **Currently Installed** tab.
3. Choose a plugin and click **Promote**.

The new plugin is now available for use in your system.

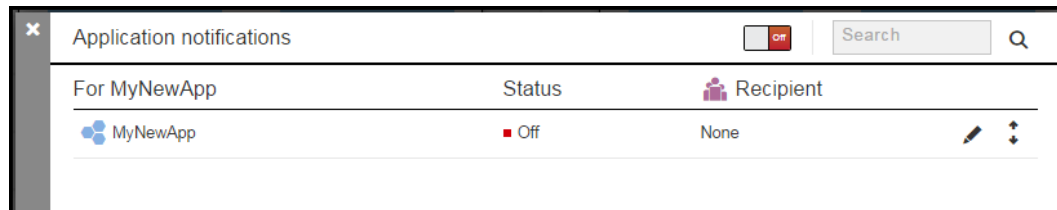
Setting Email Notifications

Review these guidelines before setting notifications:

- Configure notifications for application processes and application process steps (type: process).

When you open the Applications notifications dialog box and expand the application process details, only the application process steps (type: process) appear in the list.

- New email notifications are disabled in the application, its application processes, and the application process steps (type: process) before you configure them.



- You configure notifications in the "Application notifications / edit" dialog box.

IMPORTANT: The first time that you set notifications in this dialog box, the Notifications toggle changes to **On**. After you enter notification settings and click **OK**, email notifications are enabled at that level.

Go to [Application Notifications Dialog Box](#) on page 392 for information about how to use the "Application notifications" and "Application notifications /edit" dialog boxes.

By default, the application expects that the user creates an email configuration called "default." The email configuration defaults to the server property `/server/ec_deploy/ec_defaultEmailConfiguration`, which is set to "default."

If you want to use a different name for the email configuration, change the value of `/server/ec_deploy/ec_defaultEmailConfiguration` to the new email configuration name.

Starting from the Home page, to set email notifications:

1. Go to the Applications List.

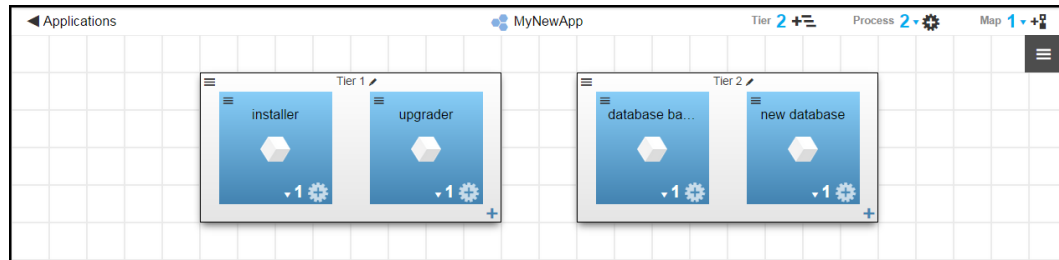
Example:

4 Applications				Select	Delete	Add
1	Arthur 23	1 Component	1 Application Process	1 Tier Map		
2	Heat Clinic Store 1.1	5 Component	2 Application Process	3 Tier Map		
3	MyNewApp	4 Component	2 Application Process	1 Tier Map		
4	Test	1 Component	1 Application Process	1 Tier Map		

2. Select an application.

The Applications Visual Editor opens.

Example:



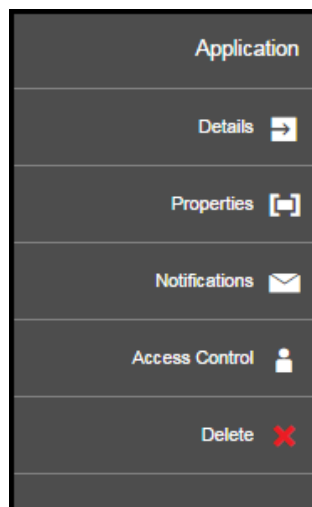
3. Click the **Menu** button.

Example:



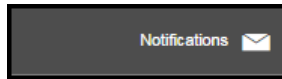
The Applications menu opens.

Example:



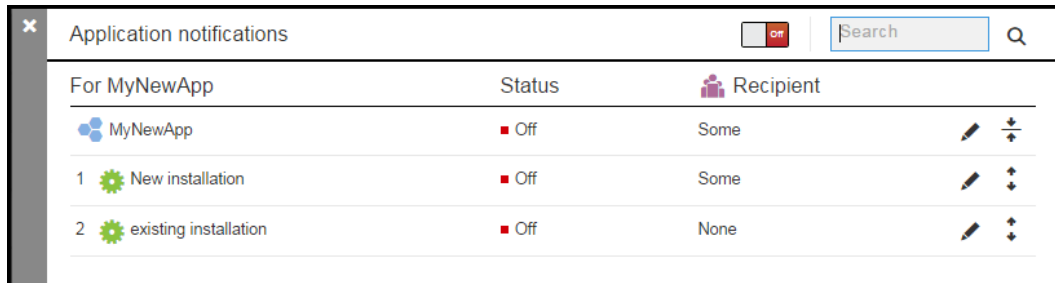
- Click **Notifications** to add a new application.

Example:



The Application notification dialog box opens.

Example:



5. Configure email notifications for the application, an application process, or a process step.

You can configure one or more notifications in an application process or other object.

Configuring recipients

In the **Who** field, you add users or groups who are configured and managed in the ElectricFlow platform or email addresses.

When you start typing a user name, group name, or email addresses, a list of names or email addresses appear that match what you are typing.

Example:

The screenshot shows the 'Application notifications / edit' interface. At the top, there is a title bar with a close button and a back arrow. Below the title bar, the notification is named 'backup' and has a toggle switch set to 'On'. The interface is divided into three main sections: 'Who', 'When', and 'Where'. The 'Who' section has a label 'Add users, groups, or email addresses:' and a text input field containing the letter 'a'. A dropdown menu is open below the input field, showing a list of suggestions: 'admin (?)', 'admin-asia (?)', 'admin-aus (?)', 'admin-uk (?)', 'admin-us (?)', 'jadams (?)', and 'sclaus (?)'. The 'When' section has a label 'Event:' and a dropdown menu showing 'Both Failed and Succes...'. The 'Where' section has a label 'Environment:' and a dropdown menu showing 'All'. There are also plus and minus icons for adding or removing sections.

If one of the suggestions matches the name or email address, select it, or continue typing. You can add more than one name or email address.

Example:

The screenshot shows the 'Application notifications / edit' interface. At the top, there is a title bar with a close button and a back arrow. Below the title bar, the notification is named 'backup' and has a toggle switch set to 'On'. The interface is divided into three main sections: 'Who', 'When', and 'Where'. The 'Who' section has a label 'Add users, groups, or email addresses:' and a text input field. Below the input field, there are several suggestions: 'admin', 'admin-asia', 'sclaus', 'jadams', 'userX@gmail.com', and 'DevT200@gmail.com'. The 'When' section has a label 'Event:' and a dropdown menu showing 'Both Failed and Succes...'. The 'Where' section has a label 'Environment:' and a dropdown menu showing 'All'. There are also plus and minus icons for adding or removing sections.

Configuring the event that triggers the notification

In the **When** field, you select the event that triggers a notification to be sent to the recipients in the **Who** field. The default is **Both Failed and Successful**. Click in the **When** field to select the event for the notification.

Example:

Application notifications / edit

backup On

Who	When	Where
Add users, groups, or email addresses:	Event:	Environment:
admin admin-asia sclaus jadams userX@gmail.com	Both Failed and Successful ▼	All
DevT200@gmail.com	Run Failed	
	Run Successful	
	Both Failed and Successful	

Configuring the environments where the notification applies

In the **Where** field, you select the environments to which the notifications apply. Click in the **Where** field to select the environments, which are the environments to which the application is mapped in the tier map.

Example:

Application notifications / edit

backup On

Who	When	Where
Add users, groups, or email addresses:	Event:	Environment:
admin admin-asia sclaus jadams userX@gmail.com	Run Successful ▼	All
DevT200@gmail.com		hc-store dev

6. Select and edit the email notification message.

7. Add another notification for the the application, an application process, or a process step.

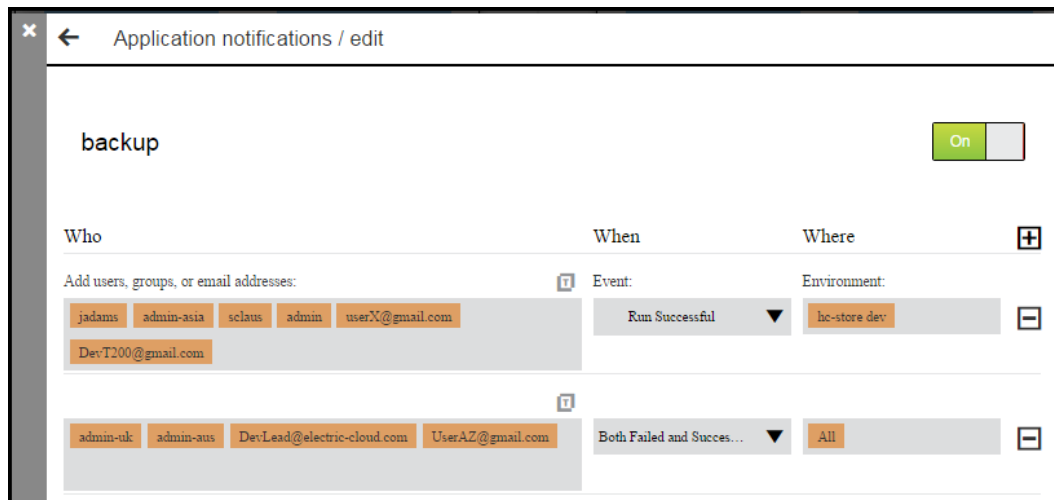
Click the **Add Notifications** button to add a new notification.

Example:



After you have added your email notifications, click **OK** to save the settings and return to the Application notifications dialog box.

Example:



The screenshot shows the 'Application notifications / edit' dialog box. At the top, there's a title bar with a close button and a back arrow. Below the title bar, the notification name 'backup' is displayed on the left, and a toggle switch labeled 'On' is on the right. The main area is divided into three columns: 'Who', 'When', and 'Where'. Under 'Who', there's a text input 'Add users, groups, or email addresses:' followed by a list of email addresses: 'jadamis', 'admin-asia', 'sclaus', 'admin', 'userX@gmail.com', and 'DevT200@gmail.com'. Under 'When', there's a dropdown menu labeled 'Event:' with 'Run Successful' selected. Under 'Where', there's a dropdown menu labeled 'Environment:' with 'bc-store dev' selected. There are also expand/collapse icons for each section. At the bottom, there's another row with email addresses 'admin-uk', 'admin-aus', 'DevLead@electric-cloud.com', and 'UserAZ@gmail.com', and dropdowns for 'Both Failed and Succes...' and 'All'.

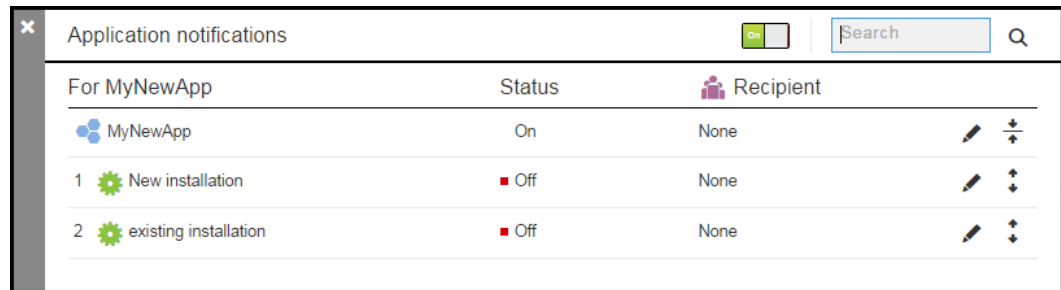
8. (Optional) Enable email notifications for the application, application processes, or process steps that are not already enabled.

To enable email notifications at the application level:

- Click the Notifications toggle and change it to **On**.

The status of the application changes to **On**.

Example:

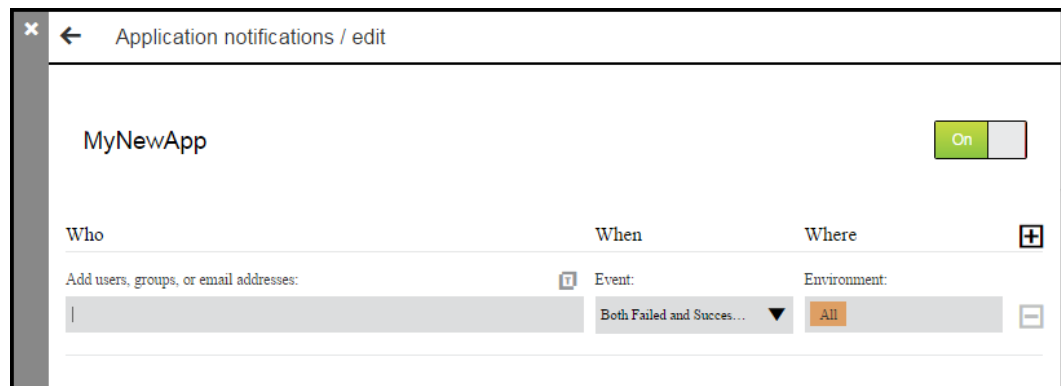


For MyNewApp	Status	Recipient
MyNewApp	On	None
1 New installation	Off	None
2 existing installation	Off	None

- Click the **Edit** button to open the **Application notifications / edit** dialog box.

The **Application notifications / edit** dialog box appears. The Notification toggle changes to **On**.

Example:



MyNewApp On

Who: Add users, groups, or email addresses:

When: Event: Both Failed and Success... ▼

Where: Environment: All [-]

To enable notifications at the application process and process step levels, go to the **Application Notifications / edit** dialog box for the specific process or process step.

The dialog box opens, and the Notifications toggle is now **On**.

Example:

When you enter notification settings in the dialog box and click **OK**, the settings are saved. The **Application notifications** dialog box appears and now shows that the application process status is **On**.

Example:

For MyNewApp	Status	Recipient	
MyNewApp	On	Some	
1 New installation	On	Some	
2 existing installation	Off	None	

Selecting and Editing Email Messages

Starting in the "Application notifications / edit" dialog box:

1. Click the **Template** button .

A drop-down box opens.

Example:

Application notifications / edit

backup On

Who	When	Where
Add users, groups, or email addresses:	Event:	Environment:
jadams admin-asia sclaus admin userX@gmail.com	Run Successful	hc-store dev
DevT200@gmail.com		
admin-uk admin-aus	Both Failed and Succes...	All

Apply email message template

Default Success notific...

Cancel OK

2. Click the down arrow to open the list of email message templates that can apply to the application.

3. Select a template.

Example:

Application notifications / edit

backup On

Who When Where

Add users, groups, or email addresses:

jadams admin-asia sclaus admin userX@gmail.com

DevT200@gmail.com

admin-uk admin-aus

Event: Run Successful

Environment: hc-store dev

Both Failed and Succes... All

Apply email message template

Default Success notific... ▼

Arthur

Arthur2

Artyoms Success notification

Default Failure notification

Default Success notification

Cancel OK

If the template is the current template applied to notification, the name of the template appears in dialog box.

Example:

Apply email message template

Default Success notific... ▼

Apply Edit

If the template is not the current template, the **Apply** and **Edit** buttons appear in the dialog box.

Example:



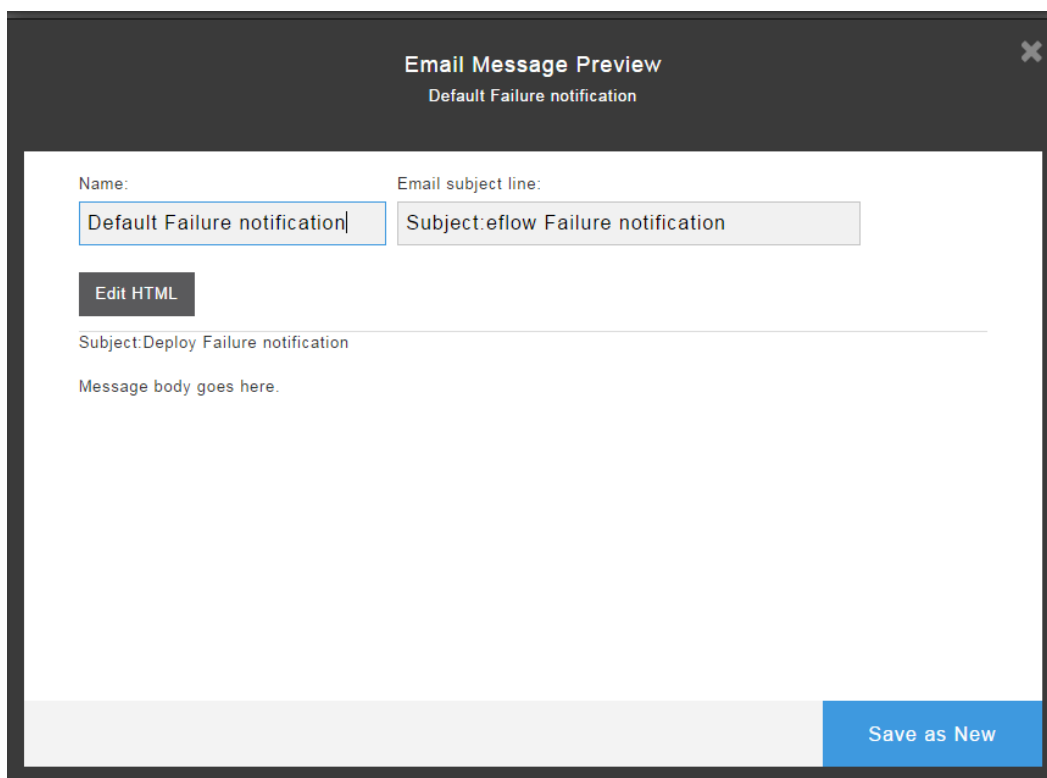
4. If you want to use the template that you selected instead of the current one and do not want to change it, skip the remaining steps in this task.
5. If you want to apply a different template or edit the template that you selected, do the remaining steps in this task.

If you click **Apply** to use the template as is, skip the remaining steps.

If you click **Edit** to modify the template to fit your needs, go to the next step.

The Email Message Preview dialog box appears.

Example:



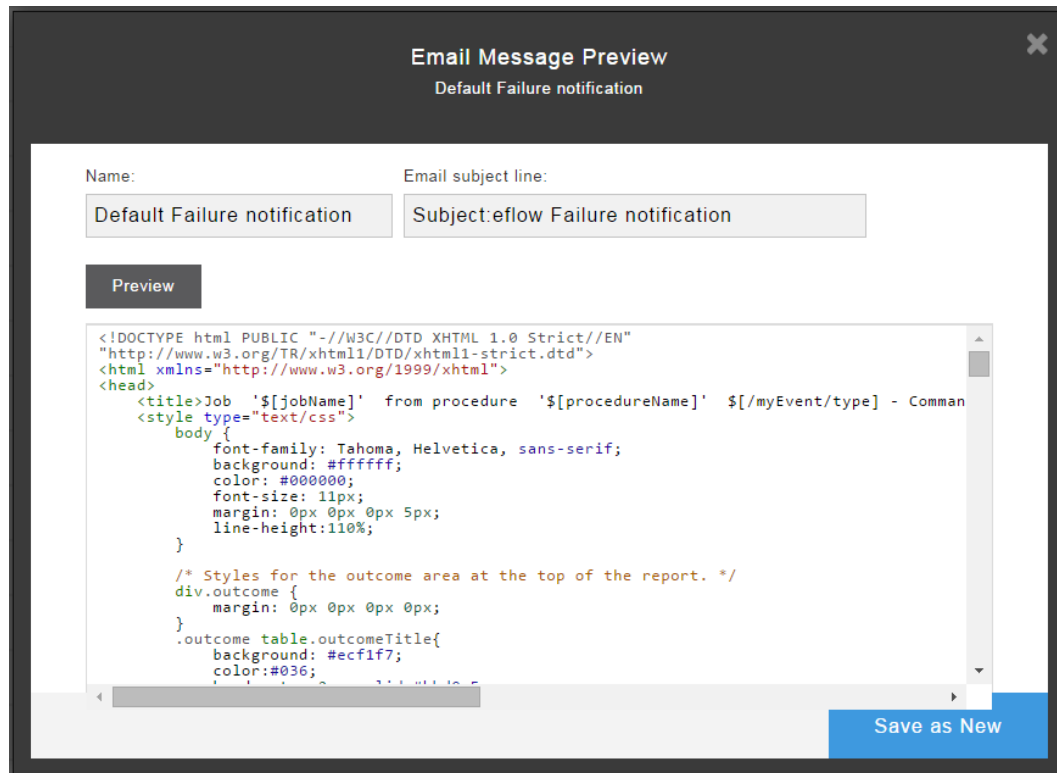
6. To edit the template:

Change the name of the template in the **Name** field.

Change the subject of the email in the **Email subject line**.

To modify the body of the email message, click **Edit HTML** and edit the HTML code.

Example:



7. Click **Preview**.

8. To save your changes:

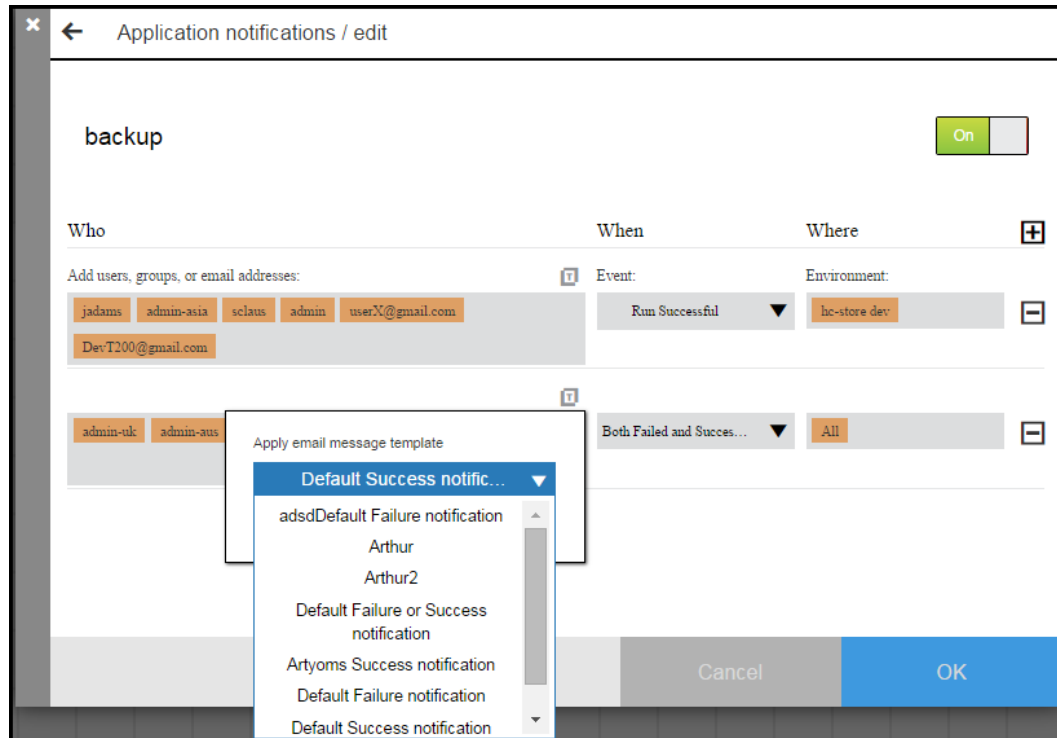
- Click **Save Changes** to save the change in an existing template
- Click **Save as New** to save the template as a new template.

The "Application notifications / edit" dialog box re-appears.

- Click the down arrow to open the list of email message templates that can apply to the application.

The new email message template is in the list.

Example:



- Click **OK** to save the settings.

Modeling Dynamic Environments

- An environment with only provisioned cloud resources.
- An environment with only static resources added to an environment template.
- An environment with provisioned cloud resources and static resources

This is the high-level process to provision cloud resources that can be dynamically spun up during the application deployment.

- Create and define one or more resource templates.

The resource template has the information required to spin up the resources on an on-demand basis and to provision dynamic resource pools. It has the following information:

- Cloud provider and cloud instance details
- Configuration management settings
- Both cloud provider and configuration management settings

2. Create and define one or more environment templates.

When modeling an environment template, you define the environment tiers and then add resources to the tiers. You can either add static resources or a resource template to an environment tier.

When adding static resources, you can select one or more resources to add to the tier.

When adding resource templates, you can select only one to add to the tier and then enter the number of resources to provision. to add dynamic resources to the tier.

Environment templates have this information:

- Name and description
- Environment tier details and properties
- Resources assigned to the environment tiers
- Cloud resource details from the resource templates

Usage Guidelines and Best Practices

This matrix shows the resource types that are allowed in environments and environment templates.

Resource Type	Origin	Environment Tier	Environment Template Tier	Use Rules
Static resources	–	Yes	Yes	<p>You may only add them to one environment. You cannot re-use them in more than one environment.</p> <p>You can add one or more static resources to an environment tier or an environment tier template.</p>
Static resource pool	–	Yes	No	<p>You may only add it to one environment. You cannot re-use it in more than one environment.</p> <p>You can add only one resource pool to an environment tier.</p>

Resource Type	Origin	Environment Tier	Environment Template Tier	Use Rules
Resource pool provisioned in the resource template	automation platform	Yes	No	You may only add it to one environment. You cannot re-use it in more than one environment. You can add only one resource pool to an environment tier.
Resource pool provisioned in an environment template	ElectricFlow	No	No	You cannot re-use the resource pools provisioned in an environment template. You can only use it in the dynamic environment created by the environment template.
Resource template	–	No	Yes	You may only add one resource template to an environment template tier.

In environments and environment templates, you can create a tier with static resources, a static resource pool, or a static resource template. When you are editing the tier, ElectricFlow maintains the initial resource-type association.

For example, if you initially add static resources to an environment tier, you can add only static resources to it. You are not allowed to add a resource this environment tier.

Creating AMIs

Before creating a resource template, you must create an Amazon Machine Image (AMI) with a pre-installed agent for the cloud provider. In ElectricFlow 5.4, Amazon EC2 and OpenStack are the supported cloud providers.

AMIs with ElectricFlow Agents

1. Install agents on ElectricFlow server node machines.

See the following sections in the [ElectricFlow Installation Guide](#) for detailed instructions:

- System Requirements and Supported Platforms > Agent Platforms
- System Requirements and Supported Platforms > Server and Agent Compatibility

- Installing ElectricFlow > Interactive Command-line Installation Method > Express Agent Command-Line
- Installing ElectricFlow > Non-Server Platform Agent Installation Method
- Configuration > Environment Proxy Server Configuration > Configuring Proxy Agents

2. Create the AMI on the Amazon EC2 or OpenStack platform.

Go to <http://aws.amazon.com/amazon-linux-ami/> for Amazon EC2.

Go to <http://docs.openstack.org/image-guide/content/index.html> for OpenStack.

3. Create a resource template in ElectricFlow.

When you set the cloud provider, you can select **Amazon** or **OpenStack** and enter information in the fields in the dialog boxes. The fields that you see depend on the cloud provider that you select.

For detailed Amazon EC2 instructions, go to [Configuring Amazon EC2 as the Cloud Provider](#).

For detailed OpenStack instructions, go to [Configuring OpenStack as the Cloud Provider](#).

AMIs with ElectricFlow Agents and Chef Configuration Management

1. Install agents on ElectricFlow server node machines.

See the following sections in the [ElectricFlow Installation Guide](#) for detailed instructions:

- System Requirements and Supported Platforms > Agent Platforms
- System Requirements and Supported Platforms > Server and Agent Compatibility
- Installing ElectricFlow > Interactive Command-line Installation Method > Express Agent Command-Line
- Installing ElectricFlow > Non-Server Platform Agent Installation Method
- Configuration > Environment Proxy Server Configuration > Configuring Proxy Agents

2. Configure a Chef server with run-lists that will be applied to your cloud resources.

Go to <http://docs.chef.io/>.

3. Create the AMI on the Amazon EC2 or OpenStack platform.

Go to <http://aws.amazon.com/amazon-linux-ami/> for Amazon EC2.

Go to <http://docs.openstack.org/image-guide/content/index.html> for OpenStack.

4. Create a resource template in ElectricFlow.

- a. When you set the cloud provider, you can select **Amazon** or **OpenStack** and enter information in the fields in the dialog boxes. The fields that you see depend on the cloud provider that you select.

For detailed Amazon EC2 instructions, go to [Configuring Amazon EC2 as the Cloud Provider](#).

For detailed OpenStack instructions, go to [Configuring OpenStack as the Cloud Provider](#).

- b. When you set the configuration management tool, select **Chef**.

For detailed instructions, go to [Configuring Chef as the Configuration Management Tool](#).

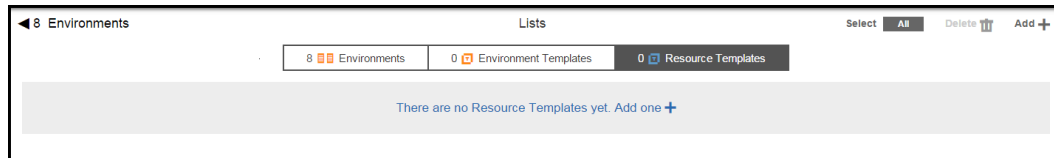
Creating Resource Templates

1. Go to the Resource Templates List.

- Starting from the Main menu, click the **Menu** button, select **Environments**, and then select **Resource Templates**.
- Starting from the Home page, click **Environments** and then click the **Resource Templates** tab.

If there are no defined resource templates, the Resource Templates List is empty.

Example:



2. Add a resource template.

- If the Resource Templates List is empty, click **There are no Resource Templates yet. Add one +** or click the **Add +** button.
- If the Resource Templates List is not empty, click the **Add +** button.
- When you click **There are no Resource Templates yet. Add one +**, the **New Resource Template** dialog box opens.

Add screen grabs here showing the configuration inputs/settings once ENGG is complete.

Example:



3. To create a new resource template:

Click **Create New**.

Enter the name of the template and an optional description of the template.

Click **Next**. The **New Cloud Provider** dialog box opens.

Go to Step 4 to Step 7 to complete the process to create a new template.

4. Enter the cloud provider settings.

5. To set Amazon EC2 as the cloud provider, go to [Configuring Amazon as the Cloud Provider](#) for detailed instructions.

IMPORTANT: Before configuring the cloud provider, you must have an Amazon Machine Image (AMI) with a pre-installed agent available.

6. To set OpenStack as the cloud provider, go to for detailed instructions.
7. (Optional) Enter the change management settings.
8. To set Chef as the configuration management tool, go to [Configuring Chef as the Configuration Management Tool](#) on page 264 for detailed instructions.

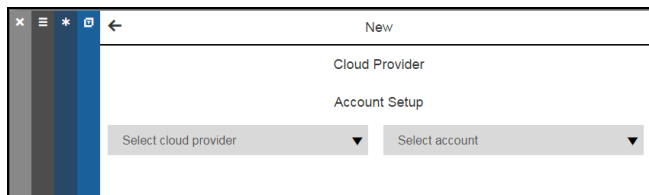
After completing all these steps, the Resource Template List now shows the templates that you created.

Configuring Amazon EC2 as the Cloud Provider

IMPORTANT: Before configuring the cloud provider, you must have an Amazon Machine Image (AMI) with a pre-installed agent available.

Starting in the **New Cloud Provider** dialog box:

Example:



1. Click **Select cloud provider**.
2. Select **Amazon**.

The **Account Setup** fields appear.

Example:

The screenshot shows a 'New' dialog box with a sidebar on the left. The main content area is titled 'New' and contains two sections: 'Cloud Provider' and 'Account Setup'. Under 'Cloud Provider', 'Amazon' is selected in a dropdown menu. To the right of this is a 'Select account' dropdown. Below these are several input fields: 'Configuration Name' (with a red error message 'Config is required'), 'Description' (containing 'EC2 integration'), 'Service URL' (containing 'https://ec2.amazonaws.com'), 'Resource Pool' (containing 'default'), and 'Workspace' (containing 'default').

The screenshot shows the same 'New' dialog box, but the 'Account Setup' section is expanded. It contains fields for 'Access ID's', 'Username' (with a red error message 'User name is required'), 'Password' (with a red error message 'Password is required'), and 'Repeat Password'. There is also a checkbox for 'Attempt Connection?' and a 'Debug Level' dropdown set to '1'. At the bottom of the dialog are 'Cancel' and 'Next' buttons.

IMPORTANT: If you have a pre-configured account with the Account Setup and Provision Parameter settings, click **Select account** to automatically enter the cloud provider settings. Then go to Step 6.

3. Enter your account settings:

- **Configuration Name**—A unique name for the EC2 connection (Required)
- **Description**—A description for this configuration. The default is *EC2 integration*.
- **Service URL**—The service URL for the EC2 service. For the Amazon public EC2, this should be *https://ec2.amazonaws.com*. (Required)
- **Resource Pool**—The name of the pool of resources on which the integration steps can run. The default is *default*.
- **Workspace**—The workspace to use for resources dynamically created by this configuration. The default is *default*.
- **Access ID's**—The access IDs, *Access ID* and *Secret Access ID*, that are required for communicating with Amazon EC2.

The configuration stores these as a credential, putting the Access ID in the **User Name** field of the credential and the Secret Access ID in the **Password** and **Retype Password** fields of the credential.

- **Attempt Connection?**—If the check box is selected, the system tries a connection to check credentials.
- **Debug Level**—The debug level for the output. The default is 1.

The possible values are

- **0** – Errors only.
- **1** – Normal headers and responses.
- **2+** – Debugging information included.

4. Click **Next**. The **Provision Parameters** fields appear.

Example:

New

Cloud Provider

Provision Parameters

Configuration: awstest

Number of Instances: 1

Group: default

Image: Image is required

Instance Type: Small(m1.small)

Key name: Keyname is required

New

Cloud Provider

Provision Parameters

Results Location:

User Data:

Zone: Zone is required

Resource Pool:

Commander Workspace:

Resource port:

New

Subnet Id:

Use Private IP for: ☒

subnet?:

Cancel Next OK

5. Enter the **Provision Parameters** values.

- **Configuration**—This field is prepopulated with the name you entered in the previous dialog box.
- **Number of Instances**—The number of instances to start. The default is 1.
- **Group**—The security group to use. The default is *default*.
- **Image**—The name of the EC2 image deploy.
- **Instance Type**—The instance type. The default is **Small(m1.small)**.

Select one of these types:

- Small(m1.small)
 - Large(m1.large)
 - Extra Large(m1.xlarge)
 - Micro(t1.micro)
 - High-Memory Extra Large(m2.xlarge)
 - High-Memory Double Extra Large(m2.2xlarge)
 - High-Memory Quadruple Extra Large(m2.4xlarge)
 - High-CPU Medium(c1.medium)
 - High-CPU Extra Large(c1.xlarge)
 - Cluster Compute Quadruple Extra Large(cc1.4xlarge)
 - Cluster GPU Quadruple Extra Large(cg1.4xlarge)
- **Key name**—The name of the key pair to use.
 - **Results Location**—Where to put the result. If the location is not specified, the result is only printed.
 - **User Data**—Extra user data to pass to the API_runInstances procedure.
 - **Zone**—The zone where the instance is created.
 - **Resource Pool**—Name of the resource pool. If you enter a name, a new resource is created and put in the pool.
 - **Commander Workspace**—Name of the workspace where the resources are created.
 - **Resource Port**—ID of the port to which the new resources are assigned.
 - **Subnet Id**—The ID of the subnet in which the instances are launched. This parameter is used with VPCs.
 - **Use Private IP for subnet?**—If this check box is selected, the subnet is Private IP network.

6. Click **Next** to set the configuration management settings.

The **New Configuration Management** dialog box opens.

Return to [Creating Resource Templates](#) on page 251 for instructions to continue creating the resource template.

7. If you do not want to set the configuration management settings, click **OK**.

The Resources Templates list now shows the resource template that you created.

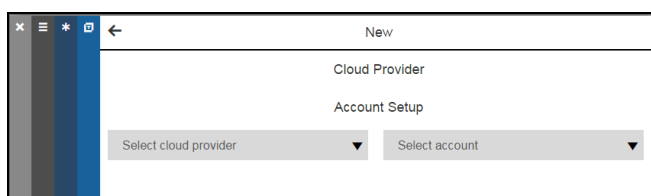
Return to [Creating Resource Templates](#) on page 251 for instructions to continue creating the resource template.

Configuring OpenStack as a Cloud Provider

IMPORTANT: Before configuring the cloud provider, you must have an Amazon Machine Image (AMI) with a pre-installed agent available.

Starting in the **New Cloud Provider** dialog box:

Example:



1. Click **Select cloud provider**.

2. Select **OpenStack**.

The **Account Setup** fields appear.

Example:

New

Cloud Provider

Account Setup

OpenStack ▼ Select account ▼

Configuration Name: Config is required

Description:

Identity Service URL: Identity service url is required

New

Cloud Provider

Account Setup

Compute Service URL: Compute service url is required

Block Storage URL: Blockstorage service url is required

Image Service URL: Image service url is required

New

Cloud Provider

Account Setup

Orchestration Service

URL:

Login as:

Username: User name is required

Password: Password is required

Repeat Password:

Compute API version:

New

Cloud Provider

Account Setup

Keystone API version: 3

Blockstorage API version: 1

Image API version: 1

Resource: local

Workspace: default

Debug Level: 1

Cancel Next

IMPORTANT: If you have a pre-configured account with the Account Setup and Provision Parameter settings, click **Select account** to automatically enter the cloud provider settings. Then go to Step 6.

3. Enter your account settings:

- **Configuration Name**—A unique name for the OpenStack configuration. (Required)
- **Description**—A description for this configuration. The default is *OpenStack configuration*.
- **Identity Service URL**—The host name or IP address of the Identity Service for OpenStack. (Required)
- **Compute Service URL**—The host name or IP address of the Compute Service for OpenStack. (Required)
- **Blockstorage URL**—The host name or IP address of the Block Storage Service for OpenStack. (Required)
- **Image Service URL**—The host name or IP address of the Image Service for OpenStack. (Required)
- **Orchestration Service URL**—The host name or IP address of the Orchestration Service for OpenStack. (Required))
- **Login as**—OpenStack account username and password. It must have enough privileges to perform API functions. (Required)
- **Compute API version**—The version of the Compute Service API. The default is 2.0. (Required)
- **Keystone API version**—The version of the Keystone Service API. The default is 3. (Required)
- **Blockstorage API version**—The version of the OpenStack block storage API. The default is 1. (Required)
- **Image API version**—The version of the Image Service API. (Required)
- **Resource**—The name of the resource or pool on which the integration steps can run. The default is *local*. (Required)
- **Workspace**—The workspace where the configuration dynamically creates resources. The default is *default*. (Required)
- **Debug Level**—The debug level for the output. The default is 1.

The possible values are

- **0** – Errors only.
- **1** – Normal headers and responses.
- **2+** – Debugging information included.

4. Click **Next**. The **Provision Parameters** fields appear.

Example:

The screenshot shows a web form titled "New" with a "Cloud Provider" section. Below this is the "Provision Parameters" section, which contains five input fields, each with a red error message indicating it is required:

- Configuration: Connection config is required
- Tenant ID: Tenant id is required
- Number of Instances:
- Image ID: Image is required
- Flavor ID: Flavor is required

The screenshot shows a web form titled "New" with a "Cloud Provider" section. Below this is the "Provision Parameters" section, which contains four input fields, each with a red error message indicating it is required:

- Key Pair Name: Key pair name is required
- Security Group(s):
- Availability Zone:
- Customization script:

5. Enter the **Provision Parameters** values.

- **Configuration**—The name of a valid existing configuration with the connection information. (Required)
- **Tenant ID**—The ID of the tenant to use. (Required)
- **Number of Instances**—The number of servers to deploy. If there is more than one, a suffix ((_#) is added to the server names. The default is 1. (Required)
- **Image ID**—The ID of an existing image in OpenStack. (Required)
- **Flavor ID**—The ID of the flavor to use. (Required)
- **Key Pair Name**—The name of a key pair to use. (Required)
- **Security Group(s)**—A list of security groups.
- **Availability Zone**—The zone where the server is launched.
- **Customization script**—Configuration information or scripts to execute when the server starts.
- **Results Location**—Where the properties are stored.
- **Resource Pool**—The resource pool name associated with the machines in this configuration when resources are created. .
- **Resource workspace**—The workspace that the resource uses.

6. Click **Next** to set the configuration management settings.

The **New Configuration Management** dialog box opens.

Return to [Creating Resource Templates](#) on page 251 for instructions to continue creating the resource template.

7. If you do not want to set the configuration management settings, click **OK**.

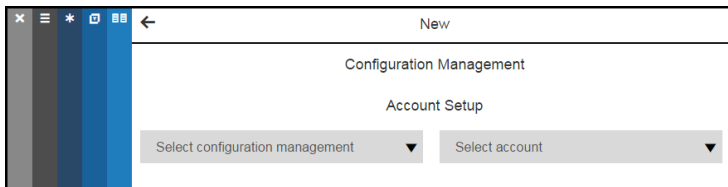
The Resources Templates list now shows the resource template that you created.

8. Return to [Creating Resource Templates](#) on page 251 for instructions to continue creating the resource template.

Configuring Chef as the Configuration Management Tool

Starting in the **New Configuration Management** dialog box:

Example:



The screenshot shows a dialog box titled "New". Inside the dialog, there are two main sections: "Configuration Management" and "Account Setup". Under "Configuration Management", there is a dropdown menu with the text "Select configuration management". Under "Account Setup", there is a dropdown menu with the text "Select account".

1. Click **Select cloud provider**.
2. Select **Amazon**.

The **Account Setup** fields appear.

Example:

New

Configuration Management

Account Setup

Chef Select account

Configuration Name: Config is required

Description: Chef configuration

Chef Server URL: Server is required

New

Configuration Management

Account Setup

Chef Server URL: Server is required

Login as:

Key Name Required

Key Required

Cancel Next

IMPORTANT: If you have a pre-configured account with the Account Setup and Converge Parameter settings, click **Select account** to automatically enter the configuration management settings. Then go to Step 6.

3. Enter your account settings:

- **Configuration Name**—A unique name for the configuration. (Required)
- **Description**—A description for this configuration. The default is *Chef configuration*.
- **Chef Server URL**—URL to the Chef server.
- **Login as**—Private key for authentication. (Required)

To log in, enter the **User Name** and private **Key**.

4. Click **Next**.

The **Converge Parameters** fields appear.

Example:

New

Configuration Management

Converge Parameters

Configuration:

Chef-client Path:

Run as a user with full ☒ system privileges:

Run List: Run list is required

New

Configuration Management

Converge Parameters

Run List: Run list is required

Node Name:

Additional Arguments:

Cancel OK

5. Enter the **Provision Parameters** values.

- **Configuration**—This field is prepopulated with the name you entered in the previous dialog box.
- **Chef-client Path**—The path to the chef-client (executable), such as /usr/bin/chef-client (the default value). (Required)
- **Run as user with full system privileges**—If this check box is selected, you run the application as a user with full system privileges.
- **Run List**—The ordered list of Chef recipes to run.
- **Node Name**—The name of the node to which recipes are added. (Required)
- **Additional Arguments**—Additional arguments use when running the application.

6. Click **OK**.

The Resource Templates List now shows the resource template that you created.

7. Return to [Creating Resource Templates](#) on page 251.

Viewing and Editing Resource Templates

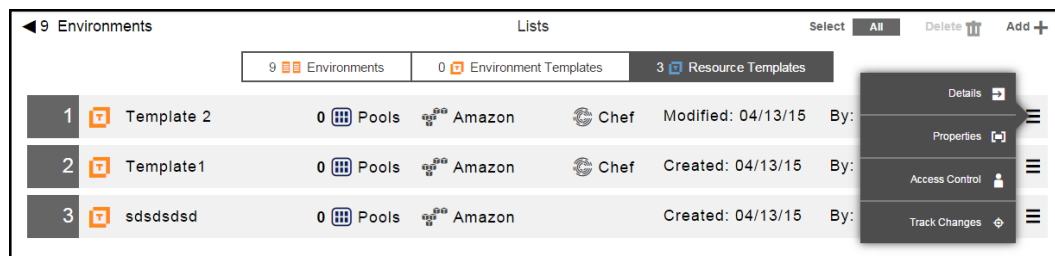
Resource Template Details

To view and edit the resource template details, starting in the Resource Template List:

1. Choose a resource template and click the **Menu** button.

A list of menu options appears.

Example:



2. Select **Details**.

The **Edit Resource Template Details** dialog box opens.

Example:

The screenshot shows a dialog box titled 'Edit' with a sub-header 'Resource Template Details'. It has five tabs: 'Details' (selected), 'Cloud Provider', 'Provision', 'Configuration', and 'Converge'. In the 'Details' tab, there are two input fields: 'Name' with the value 'Template1' and 'Description' with the value 'Description'.

3. In the **Details** tab, edit the name and description of the resource template.

4. In the **Cloud Provider** tab, edit the cloud provider account information.

Example:

The screenshot shows the same dialog box, but now the 'Cloud Provider' tab is selected. At the top, there is a dropdown menu showing 'Amazon'. Below it, there are five input fields with labels and icons: 'Configuration Name' (value: 'awstest'), 'Description' (value: 'EC2 integration'), 'Service URL' (value: 'https://ec2.amazonaws.com'), 'Resource Pool' (value: 'default'), and 'Workspace' (value: 'default'). At the bottom, there are three buttons: 'Cancel', 'Next', and 'OK'.

5. In the **Provision** tab, edit the information about a provision settings for the cloud provider.

Example:

The screenshot shows the 'Edit' dialog box with the 'Provision' tab selected. The 'Resource Template Details' section contains the following fields:

Field	Value
Configuration	awstest
Number of Instances	1
Group	default
Image	ec2test
Instance Type	Small(m1.small)
Key name	asasasaa

At the bottom of the dialog are three buttons: 'Cancel', 'Next', and 'OK'.

6. In the **Configuration** tab, view and edit the configuration management details.

Example:

The screenshot shows the 'Edit' dialog box with the 'Configuration' tab selected. The 'Resource Template Details' section contains the following fields:

Field	Value
Configuration Name	chef config
Description	Chef configuration
Chef Server URL	www.xxx.com

At the top of the configuration section is a dropdown menu currently set to 'Chef'. At the bottom of the dialog are three buttons: 'Cancel', 'Next', and 'OK'.

7. In the **Converge** tab, view and edit the details about how the virtual instances are converged in the defined configuration.

Example:

The screenshot shows a dialog box titled "Edit" with a sub-header "Resource Template Details". It contains five tabs: "Details", "Cloud Provider", "Provision", "Configuration", and "Converge". The "Amazon" cloud provider is selected in a dropdown menu. Below the menu, there are five labeled input fields: "Configuration Name" with the value "awstest", "Description" with "EC2 Integration", "Service URL" with "https://ec2.amazonaws.com", "Resource Pool" with "default", and "Workspace" with "default". At the bottom of the dialog are three buttons: "Cancel", "Next", and "OK".

8. Click **Next** to go to the next tab.
9. Click **OK** to save your changes.

Resource Template Properties

To view and edit the resource template properties, starting in the Resource Template List:

1. Choose a resource template and click the **Menu** button. A list of menu options appears.
2. Select **Properties**.

The **Properties** dialog box opens.

Example:

The screenshot shows a "Properties" dialog box. At the top, it says "0 Properties" and has buttons for "Delete", "Folder", and "Add". Below this is a table with two columns: "Name" and "Value". The table contains one row with a folder icon and the name "ffffff". To the right of the table is a "Select" dropdown menu with "All" selected.

3. To view more details about a property, click the **Expand** button next to the property name.

4. To add a property:

- a. Click the **Add +** button.

The **Property Details** dialog box opens.

Example:

- b. Enter the details and click **OK** to save them.

The **Property Details** dialog box now shows the new property that you added.

Example:

Name	Value
fffff	Select All
ddd	foo

Access Control Settings

When you click **Access Control**, you go to the Access Control page in the automation platform.

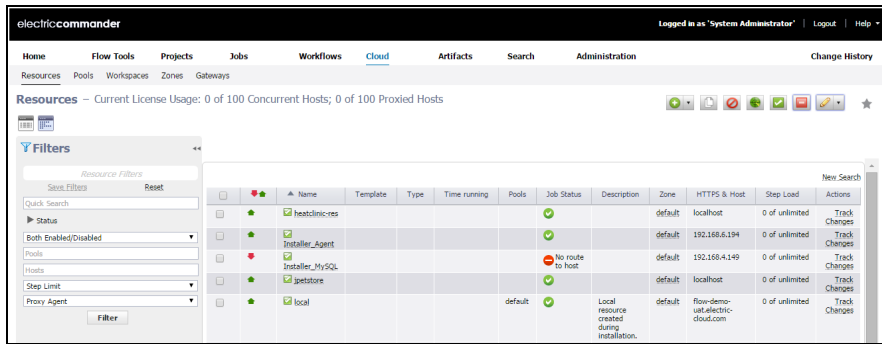
Change Tracking

When you click **Track Changes**, the Change History for the resource template opens.

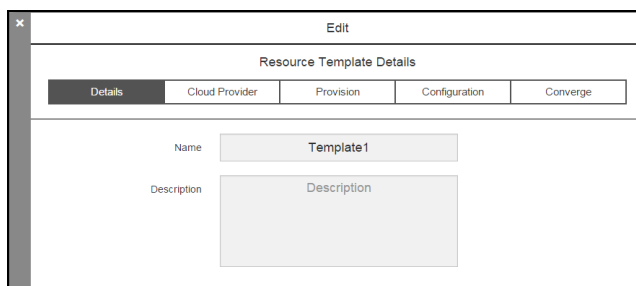
Resources Page

How to get here: Click the **Main Menu** button > **Automations** > **Resources**.

In the **Resources** page, click the name of the resource template to view the resource template details.



The **Edit Resource Template Details** dialog box opens:

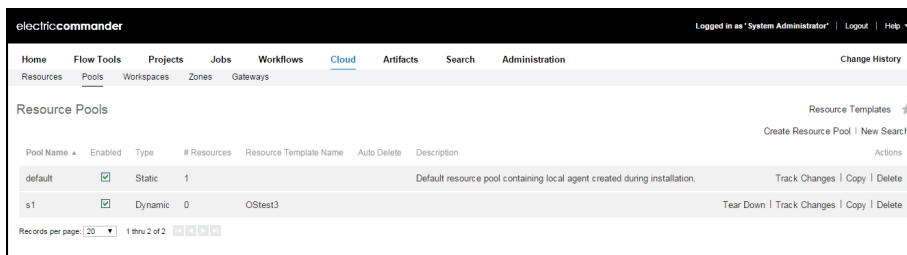


You can view and edit the resource template details as described in the previous sections.

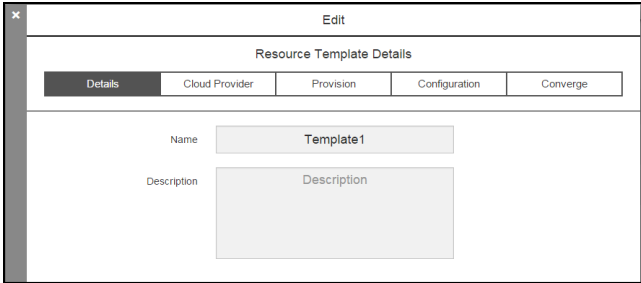
Resource Pools Page

How to get here: Go to the automation platform Help > **Cloud** tab > select **Pools**.

In the **Resource Pools** page, you get information about the resource pools. The page lists static and dynamic resource pools. Static resource pools are created in the automation platform. Dynamic resource pools are created by provisioning a resource template in ElectricFlow or in the automation platform.



When you click the name of the resource template that provisioned the dynamic resource pool, the **Edit Resource Template Details** dialog box opens, where you can view the details.



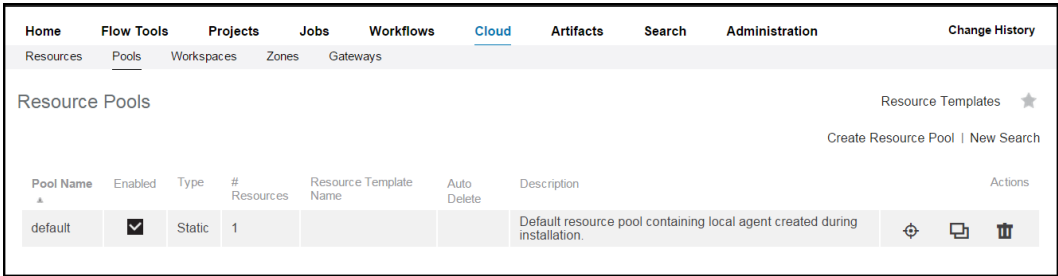
You can view and edit the resource template details as described in the previous sections.

Accessing the Resource Templates in the Automation Platform

In automation platform UI:

- 1. Go to **Cloud > Pools**.

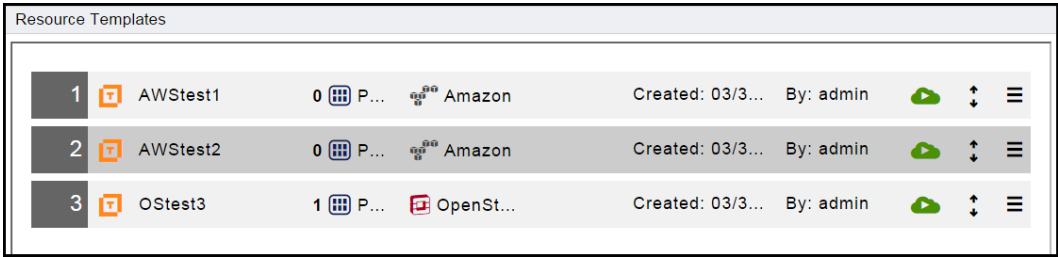
Example:




- 2. Click **Resource Templates**.

The **Resource Templates** list opens.

Example:



- 3. (Optional) To view details about a resource template, choose a template and click the **Menu** button () for it.

The Context menu opens.

4. Click one of these options:

- **Details**—The **Edit Resource Template Details** dialog box opens.

The tabs that appear in the dialog box depend on the cloud provider specified in the resource template.

- **Properties**—The **Properties** dialog box opens.

You can view and edit the properties that apply to the resource template.

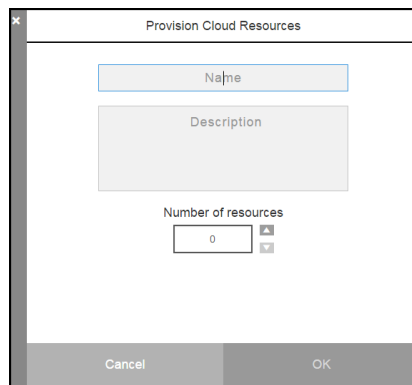
- **Access Control**—The Access Control page opens.

- **Track Changes**—The Change History for the resource template opens.

5. To provision the resource template, choose a resource template and click the **Provision** button.

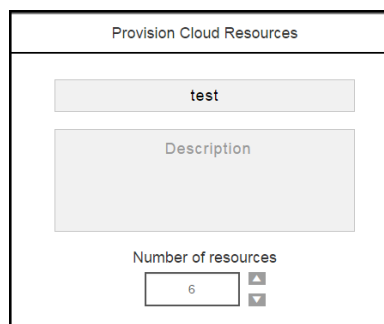
The **Provision Cloud Resources** dialog box opens.

Example:



6. Enter the resource pool name, an optional description, and the number of resources in the pool, and then click **OK**.

Example:



You go to the Job Details page in the automation platform.

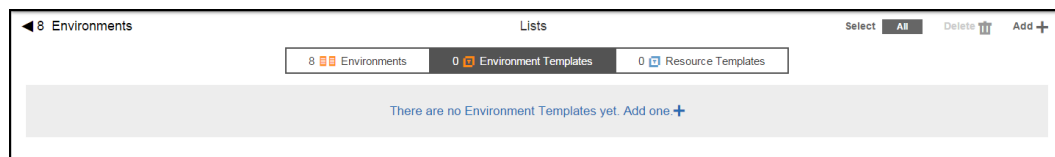
Creating Environment Templates

Creating a New Environment Template

1. Go to the Environment Templates List.
 - Starting from the Main menu, click the **Menu** button, select **Environments**, and then select **Environment Templates**.
 - Starting from the Home page, click **Environments** and then click the **Environment Templates** tab.

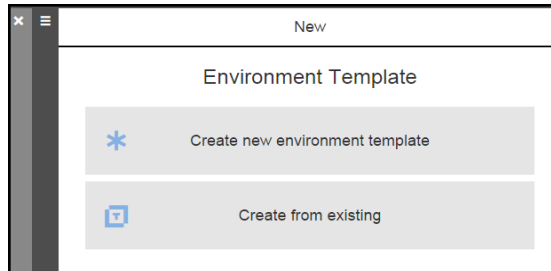
If there are no defined resource templates, the Resource Templates List is empty.

Example:



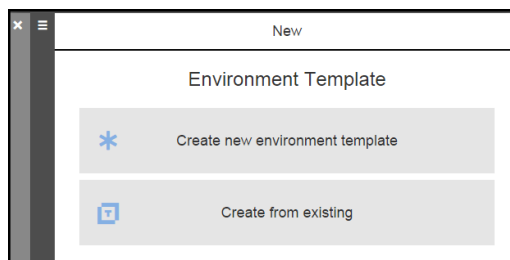
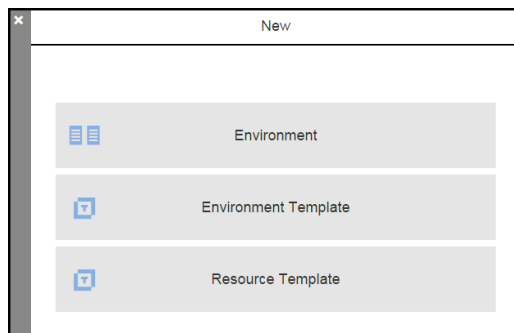
2. Add a resource template.
 - a. If the Resource Templates List is empty, click **There are no Environment Templates yet. Add one +** or click the **Add +** button.
 - b. If the resource list is not empty, click the **Add +** button.
 - c. When you click **There are no Environment Templates yet. Add one +**, the **New Environment Template** dialog box opens.

Example:



- d. After you click the **Add +** button, click **Environment Template** in the **New** dialog box.
- The **New Environment Template** dialog box opens.

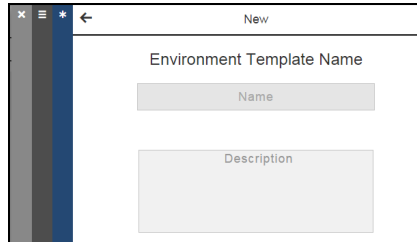
Example:



3. Click **Create new environment template** to create a template with new settings.

The **New** dialog box opens.

Example:

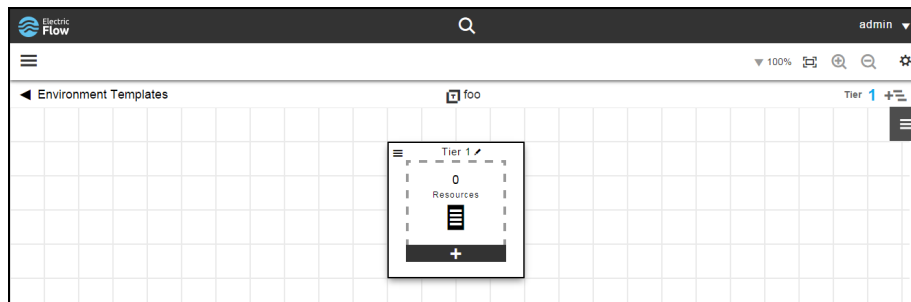


The 'New' dialog box is shown with a title bar containing a close button, a maximize button, and a back arrow. The main content area has a label 'Environment Template Name' above a text input field labeled 'Name'. Below this is a larger text area labeled 'Description'.

4. Enter a name and an optional description, and click **OK**.

The Environment Templates Visual Editor opens. The name of the environment template is at the top of the page. There is one tier with no assigned resources.

Example:



5. Click the **+** button to add a resource to the tier.

The **New** dialog box to add resources opens.

Example:



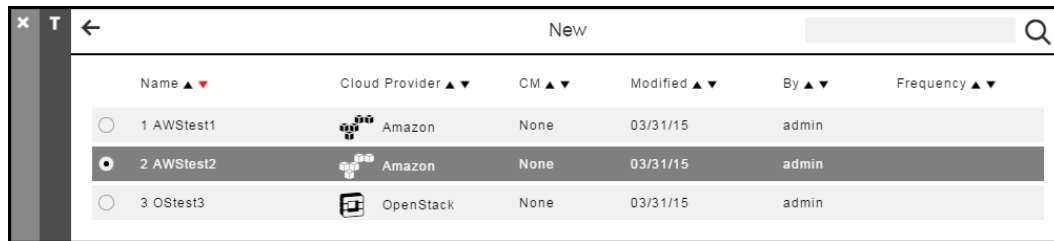
The 'New' dialog box is shown with a title bar containing a close button and a back arrow. The main content area has two buttons: 'Add resources' with a list icon and 'Add resource template' with a document icon.

6. Click **Add resource template** to select a resource template.

The Resource Templates List opens.

7. Select a resource template and click **OK**.

Example:

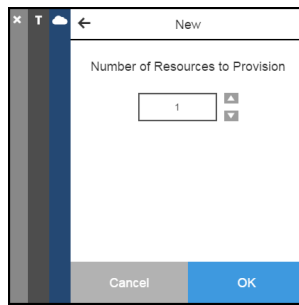


The screenshot shows a 'New' dialog box with a table of resource templates. The table has columns for Name, Cloud Provider, CM, Modified, By, and Frequency. Three templates are listed: 1 AWStest1 (Amazon), 2 AWStest2 (Amazon), and 3 OStest3 (OpenStack). The second template, 2 AWStest2, is selected with a radio button.

	Name ▲ ▼	Cloud Provider ▲ ▼	CM ▲ ▼	Modified ▲ ▼	By ▲ ▼	Frequency ▲ ▼
<input type="radio"/>	1 AWStest1	Amazon	None	03/31/15	admin	
<input checked="" type="radio"/>	2 AWStest2	Amazon	None	03/31/15	admin	
<input type="radio"/>	3 OStest3	OpenStack	None	03/31/15	admin	

The **New** dialog box to select the number of resources to provision opens.

Example:



The screenshot shows the 'New' dialog box with the 'Number of Resources to Provision' field set to 1. The dialog has 'Cancel' and 'OK' buttons at the bottom.

Number of Resources to Provision

1

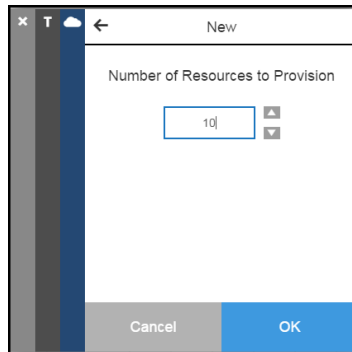
Cancel OK

8. Enter the number of resources to provision and click **OK**.

You must provision at least one resource (1).

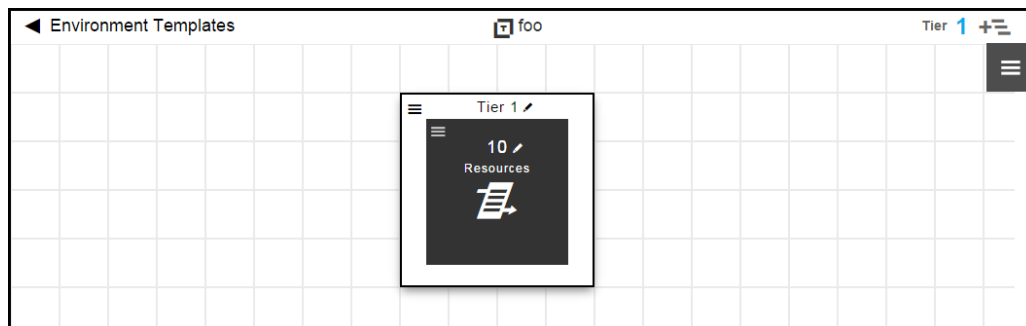
Note: You cannot override the number of resources when deploying an application with an environment template.

Example:



The Environment Templates Visual Editor now shows that the tier (Tier 1) has 10 dynamic cloud resources set to be provisioned when the application is deployed.

Example:



9. To add static resources to an environment template:

- a. Click the **Add tier** button to create a new tier.

A new environment tier appears in the Environment Templates Visual Editor.

- b. Click the **+** button in the new tier to add resources to it.

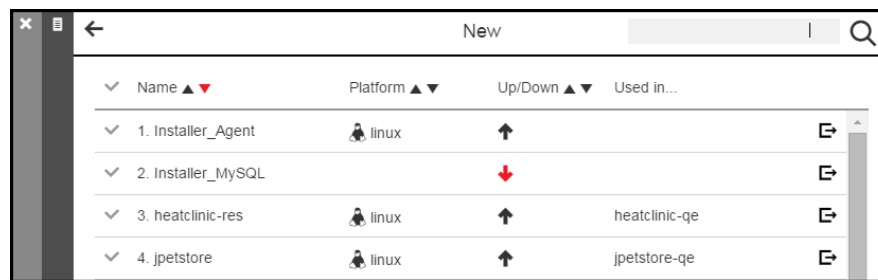
The **New** dialog box opens.

- c. Click **Add resources** to add static resources to the environment tier.

A list of static resources opens.

Note: You cannot override the setting for the number of static resources when deploying the application with an environment template

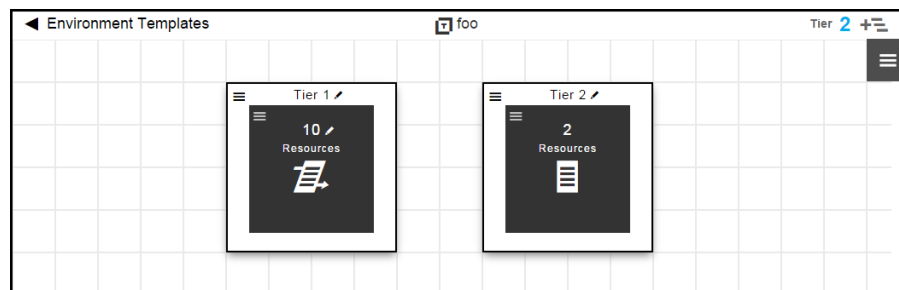
Example:



- d. Select one or more resources in the list, and click **OK**.

The Environment Templates Visual Editor now shows that Tier 2 has two static resources.

Example:



Creating an Environment Template Based on an Existing Template

1. Go to the Environment Templates List.

- Starting from the Main menu, click the **Menu** button, select **Environments**, and then select **Environment Templates**.
- Starting from the Home page, click **Environments** and then click the **Environment Templates** tab.

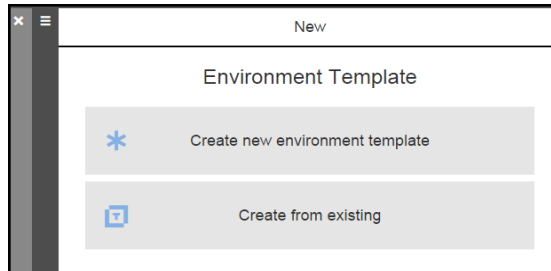
2. Add a resource template.

If the Resource Templates List is empty, click **There are no Environment Templates yet. Add one +** or click the **Add +** button.

If the resource list is not empty, click the **Add +** button.

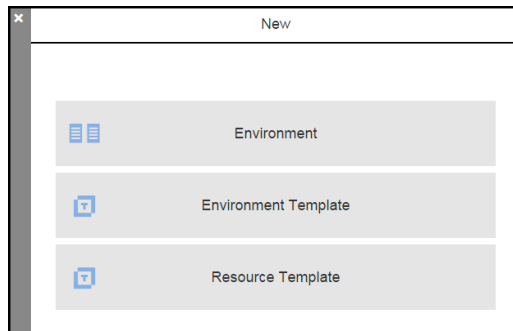
When you click **There are no Environment Templates yet. Add one +**, the **New Environment Template** dialog box opens.

Example:



After you click the **Add +** button, click **Environment Template** in the **New** dialog box. The **New Environment Template** dialog box opens.

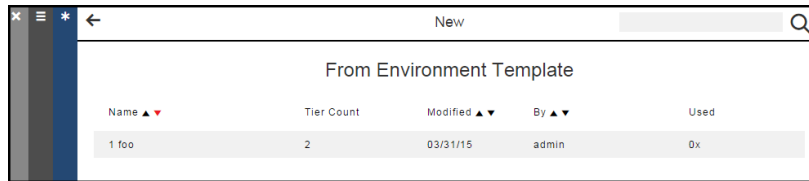
Example:



- Click **Create from existing** to create a template based on an existing one.

The list of existing templates opens.

Example:



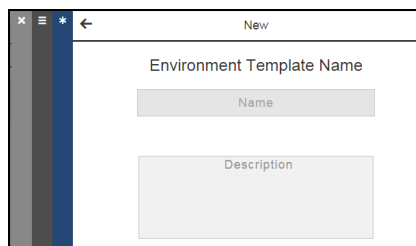
The screenshot shows a 'New' dialog box with a search bar and a table titled 'From Environment Template'. The table has columns for Name, Tier Count, Modified, By, and Used. One row is visible with the name 'foo', tier count '2', modified date '03/31/15', user 'admin', and usage '0x'.

Name ▲▼	Tier Count	Modified ▲▼	By ▲▼	Used
1 foo	2	03/31/15	admin	0x

- Select an environment template.

The **New** dialog box opens.

Example:



The screenshot shows a 'New' dialog box with two input fields: 'Environment Template Name' and 'Description'. The 'Name' field is currently empty.

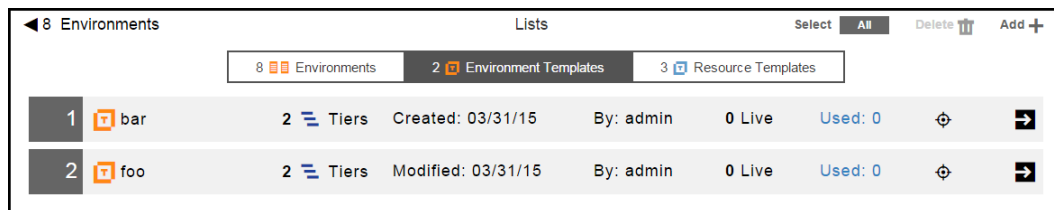
Viewing and Editing Environment Templates

Environment Template Details

To view and edit the environment template details, starting in the Environment Template List:

- Choose an environment template and click the **View Details** button.

Example:

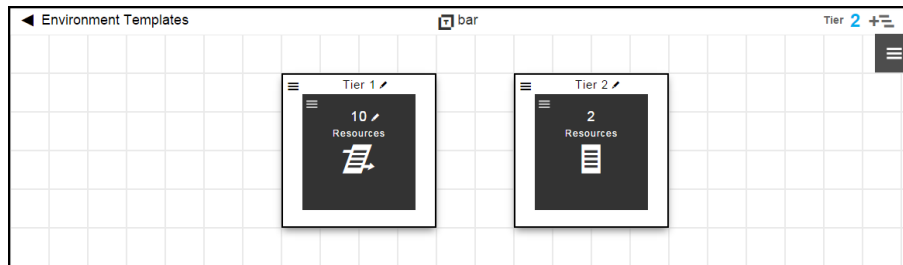


The screenshot shows a list of environments. At the top, there are tabs for '8 Environments', '2 Environment Templates', and '3 Resource Templates'. The 'Environment Templates' tab is selected. Below the tabs, there is a table with two rows. Each row has a number, a name, a tier count, creation/modification date, user, live count, used count, and a 'View Details' button.

8 Environments								Lists		Select	All	Delete	Add +
<div>8 Environments 2 Environment Templates 3 Resource Templates</div>													
1	bar	2 Tiers	Created: 03/31/15	By: admin	0 Live	Used: 0							
2	foo	2 Tiers	Modified: 03/31/15	By: admin	0 Live	Used: 0							

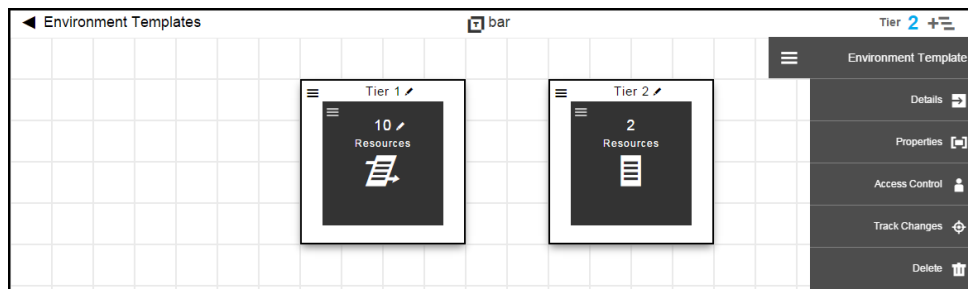
The Environment Templates for the selected template opens.

Example:



2. Click to **Menu** button to open the context menu for the environment template.

Example:



Click **Details** to view or edit the name and description of the environment template.

Click **Properties** to view or edit the environment template properties.

Click **Access Control** to go to the Access Control page in the automation platform.

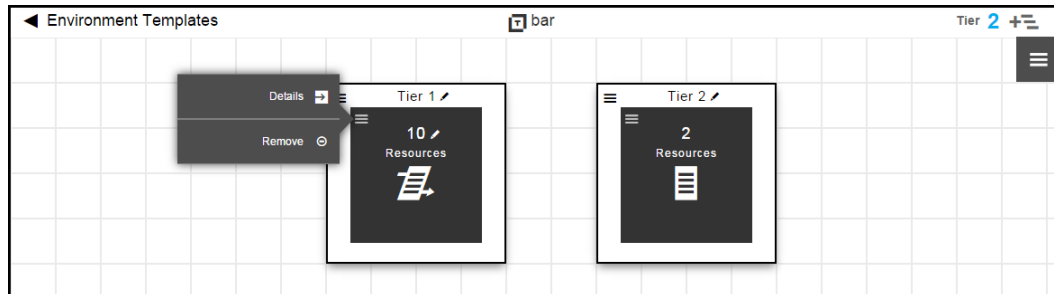
Click **Track Changes** to open the Change History of the environment template.

Click **Delete** to delete this template.

3. In an environment tier with dynamic resources, click the **Menu** button for the resources.

The context menu opens.

Example:



4. Click **Details**. The list of resource templates opens.

The template being used is selected.

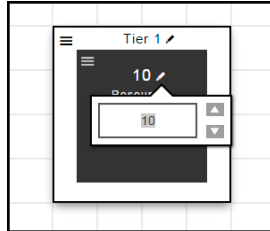
Example:

Edit						
	Name ▲ ▼	Cloud Provider ▲ ▼	CM ▲ ▼	Modified ▲ ▼	By ▲ ▼	Frequency ▲ ▼
<input type="radio"/>	1 AWSTest1	Amazon	None	03/31/15	admin	
<input checked="" type="radio"/>	2 AWSTest2	Amazon	None	03/31/15	admin	
<input type="radio"/>	3 OSTest3	OpenStack	None	03/31/15	admin	

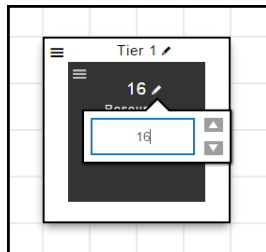
5. Click the **Edit** button in an environment tier with dynamic resources.

The number box opens and shows the number of provisioned dynamic resources. In the example, there are 10 dynamic resources.

6. To change the number of provisioned dynamic resources:
 - a. Click the **Edit** button in an environment tier with dynamic resources.

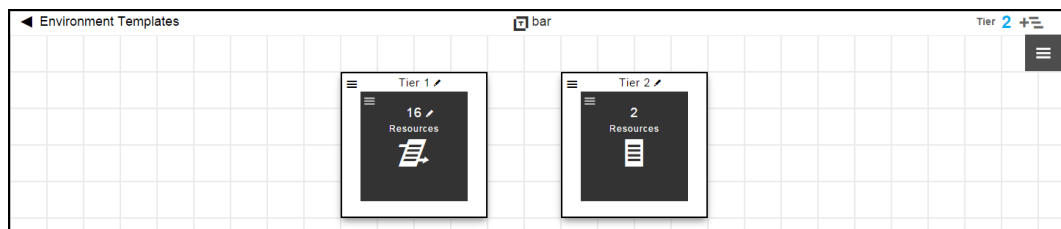
Example:

- b. Enter a new number.
Use the up and down arrows to change the number.
You must enter one or more dynamic resources.

Example:

As you change the number of resources in the number box, the visual editor automatically updates the environment tier.

- c. Click outside of the tier or press **Enter** to save the change.
The Environment Templates Visual Editor now shows the new number of dynamic resources for the tier.

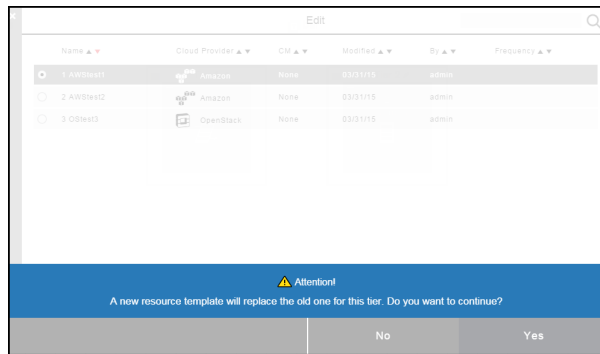
Example:

7. To change the resource template being used:

- a. Click the **Menu** button for the resources in an environment tier with dynamic resources.
- b. Click **Details** in the context menu. The list of resource templates opens.
- c. Select a different template other than the one that is currently selected.

A message appears.

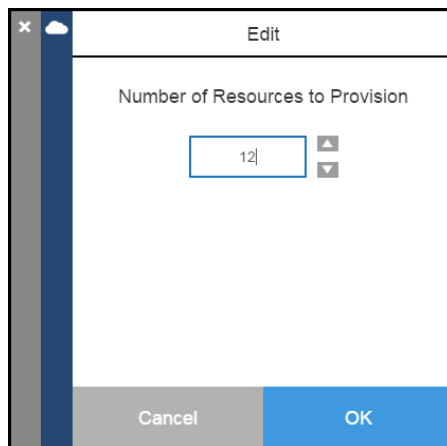
Example:



- d. Click **Yes** to replace the current template with the one that you selected.

The **Edit** dialog box to select the number of resources to provision opens.

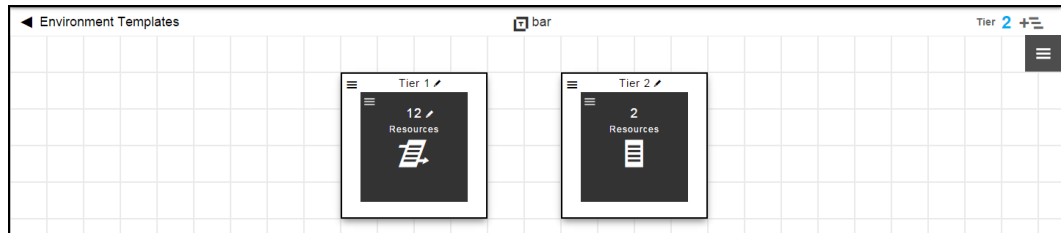
Example:



- e. To edit the number of dynamic resources to provision, enter the number and click **OK**.

The Environment Templates Visual Editor now shows that tier has the new number of dynamic resources.

Example:



Modeling Environments with Resources or Resource Pools

1. Go to the Environment List.
 - Starting from the Main menu, click the **Menu** button, select **Environments**, and then select **Environments**.
 - Starting from the Home page, click **Environments**.

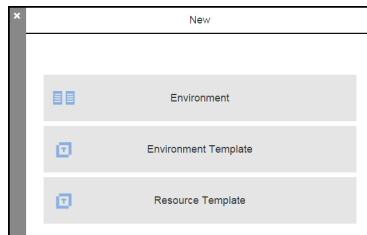
- Click the **Add +** button to add an environment.

Example:

8 Environments						2 Environment Templates	3 Resource Templates
1	hc-store dev	3 Tiers	3 Resources	1 Applications installed			
2	hc-store prod	3 Tiers	8 Resources	0 Applications installed			
3	hc-store qe	3 Tiers	6 Resources	0 Applications installed			
4	heatclinic-qe	2 Tiers	2 Resources	0 Applications installed			

The **New** dialog box opens.

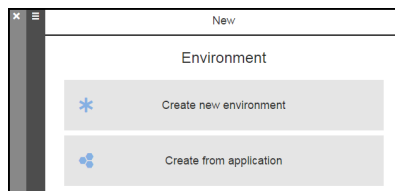
Example:



- Click **Environment**.

The **New Environment** dialog box opens.

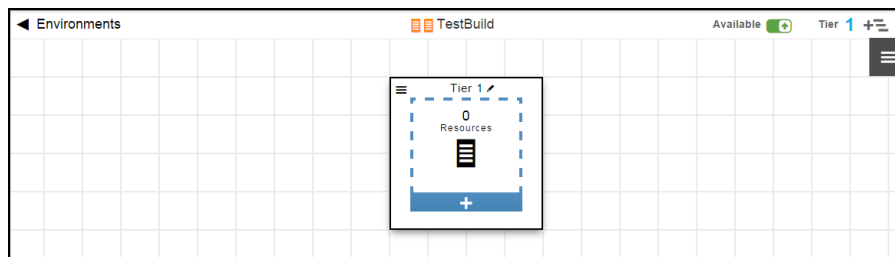
Example:



- Select **Create new environment**.
- Enter the name of the environment and an optional description, and then click **OK**.

6. Click the **+** button to add resources to the environment tier.

Example:



The **New** dialog box to add resources or a resource pool to the environment tier.

Example:



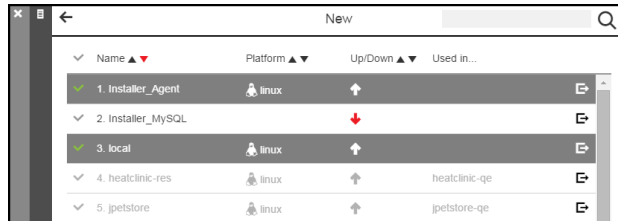
7. To add one or more static resources:

- a. Click **Add resources**.

A list of resources appears.

- b. Select one or more resources, and then click **OK**.

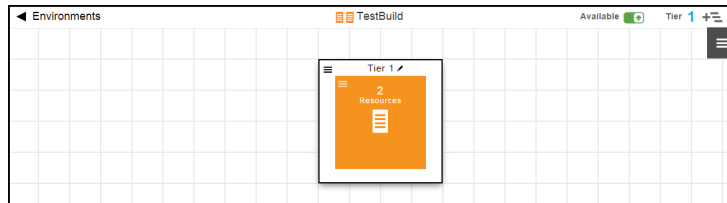
Example:



Name ▲▼	Platform ▲▼	Up/Down ▲▼	Used in...
1. Installer_Agent	linux	↑	
2. Installer_MySQL	linux	↓	
3. local	linux	↑	
4. heatclinic-res	linux	↑	heatclinic-ge
5. jpetstore	linux	↑	jpetstore-ge

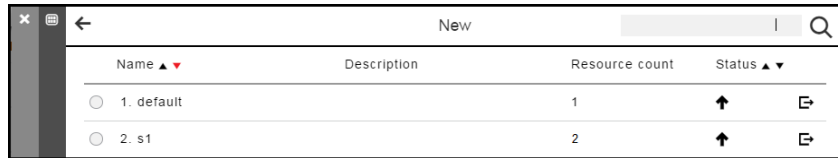
The Environments Visual Editor now shows a tier with two static resources.

Example:



8. To add a resource pool:
 - a. Click **Add resource pool**.
A list of resource pools appears.
 - b. Select one resource pool, and then click **OK**.

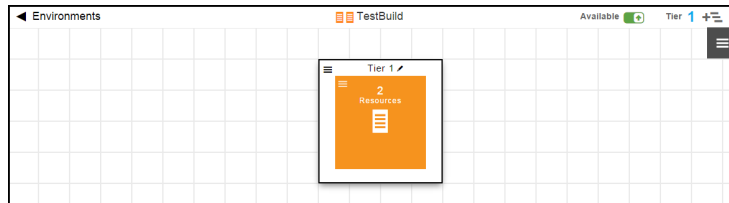
Example:



Name ▲ ▼	Description	Resource count	Status ▲ ▼
1. default		1	↑
2. s1		2	↑

The Environments Visual Editor now shows a tier with the resource pool that has two resources.

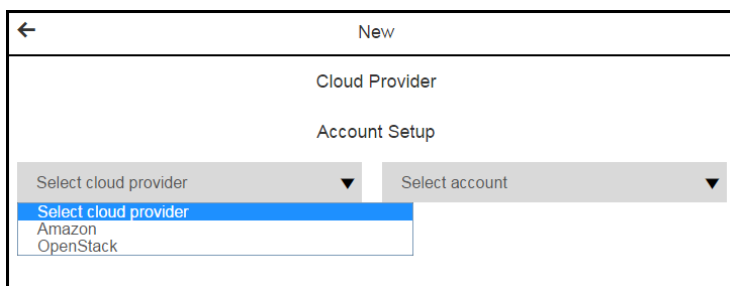
Example:



Developer Task: Creating Custom Plugins

ElectricFlow uses third-party plugins to provision cloud resources in resource templates for Dynamic Environments. Amazon EC2 (EC-EC2) and Openstack (EC-OpenStack) are supported as out-of-the-box cloud provider plugins, and Chef (EC-Chef) is supported as an out-of-the-box configuration management plugin.

To view the supported cloud provider plugins, open the **New Cloud Provider** dialog box, and click **Select cloud provider** to view the drop-down menu of cloud providers.



To view the supported configuration management plugins, click **Select configuration management** to view the drop-down menu of configuration management tools.

The screenshot shows a mobile application interface for setting up configuration management. At the top is a 'New' header with a back arrow. Below it is the 'Configuration Management' section, which contains an 'Account Setup' subsection. There are two dropdown menus side-by-side: 'Select configuration management' and 'Select account'. The 'Select configuration management' dropdown is open, showing a list with 'Select configuration management' at the top (highlighted in blue) and 'Chef' below it.

If you have an existing procedure that you would prefer to use instead of an out-of-the-box third-party plugin to provision cloud resources, you can create a custom plugin based on your deployment scenario.

The process to create custom cloud provider and configuration management plugins uses metadata to loosely couple them to the application you want to deploy. The custom plugins define specific known properties that are automatically recognized by Dynamic Environments.

To create your own plugin for other cloud platforms, go to <https://electric-cloud.github.io/index.html> to get started. This website describes how to create open source plugins and has this information:

- For an introduction to the process of building and developing open source plugins, go to the [Quick Start](#).
- To build and deploy your plugin, go to [Build and Deploy](#).
- For more information, go to [More information](#).
- Go to [Creating Custom Cloud Provider Plugins](#) on page 317 for the details to create cloud provider plugins.
- Go to [Creating Custom Configuration Management Plugins](#) on page 321 for the details to create configuration management plugins.

Creating Custom Cloud Provider Plugins

This section applies to developers who want to create custom cloud provider plugins for Dynamic Environments in ElectricFlow.

How to Create a Custom Cloud Provider Plugin

1. To convert your procedure to a plugin, define the name of the plugin and the ElectricFlow project to which the plugin belongs.

Later in this procedure, you will use these properties to create pre- and post-hooks to the plugin in ElectricFlow.

2. Define the following properties in your plugin under a top-level plugin property called `ec_cloudprovisioning_plugin`, which the Dynamic Environment system can access.

Property Name	Description
<code>ec_cloudprovisioning_plugin/</code>	Top-level plugin property directory
Properties defined by the custom plugin under <code>ec_cloudprovisioning_plugin</code>	
<code>displayName</code>	<p>Name of the plugin that appears in the New Cloud Provider dialog box.</p> <p>Example: Amazon for the EC-EC2 plugin</p>
<code>hasConfiguration</code>	<p><Boolean flag - 0 1 true false></p> <ul style="list-style-type: none"> When this property is set to 0, the plugin does not have any configuration procedures (<code>CreateConfiguration</code> and <code>DeleteConfiguration</code>). When this property is set to 1, the plugin has configuration procedures.
<code>configurationLocation</code>	<p>Name of the property sheet used by the plugin to store the saved configurations.</p> <p>This value is relative to the plugin's top-level properties.</p> <p>If this value is set as <code>ec2_cfgs</code>, the configurations are in <code>/plugins/<PLUGIN_KEY>/project/ec2_cfgs</code>.</p>
<code>operations/</code>	Property sheet for the specific operations required by the Dynamic Environment system.
Properties under <code>operations/</code> – These operations are mapped to the plugin operations through the following child properties.	
<code>createConfiguration/</code>	Properties for the procedure that creates the plugin configuration. It is usually called <code>createConfiguration</code> .
<code>deleteConfiguration/</code>	Properties for the procedure that deletes the plugin configuration. It is usually called <code>deleteConfiguration</code> .
<code>provision/</code>	Properties for the procedure that provisions virtual instances.
<code>retireResource/</code>	Properties for the procedure that tears down a previously provisioned virtual instance backing the specified resource.

Property Name	Description
<code>retireResourcePool/</code>	Properties for the procedure that tears down all previously provisioned virtual instances for the specified resource pool.
Properties for the previously listed operations defined in the <code>operations</code> property sheet	
<code>procedureName</code>	Name of the procedure name in the plugin to which the operation is mapped.
<code>ui_formRefs/</code>	Property sheet that references the <code>ui_forms</code> properties defined under the plugin's top-level properties.
<code>parameterRefs/</code>	Property sheet for the input parameters to the procedure that can be used by the Dynamic Environment system. The parameter list is operation-specific.

Go to [Example: Property Structure for a Cloud Provider Plugin](#) on page 296 for an example of the properties for the EC-EC2 (Amazon) plugin.

3. Save the plugin file in the appropriate location on your ElectricFlow server.
4. To import the plugin file to the automation platform:
 - a. In the automation platform, go to **Administration > Plugins** to open the **Plugin Manager** page.
 - b. Click the **Install from File/URL** tab.
 - c. In the **File Install** field, click **Choose file** to select the plugin file.
 - d. Click **Upload** to install it.

The plugin file appears in the **Currently Installed** tab.

 - e. Find your plugin and click **Promote** in the Actions column to make it available for use by ElectricFlow.
5. In ElectricFlow, to create a resource template:
 - a. Go to [Creating Resource Templates](#) on page 251 for the details.
 - b. In the **Select cloud provider** field, select your plugin as the cloud provider.
 - c. Enter the plugin settings in the form.
 - d. (Optional) Set the configuration management tool.

Go to [Creating Resource Templates](#) on page 251 for the details.

6. In the ectool API, enter the following commands to set pre- and post-hooks linking the plugin to an application that you later deploy in a dynamic environment.

- a. To create a pre-hook, enter

```
ectool createHook postConfigurationHook --hookType POST_CONFIGURATION --
procedureName <hookProcedure> --procedureProjectName <hookProject> --
resourceTemplateName <awsResourceTemplate> --projectName default
```

where `hookProcedure` is the your plugin name, `hookProject` is the name of the project to which the plugin belongs, and `awsResourceTemplate` is the name of the resource template that you created.

- b. To create a post-hook, enter

```
ectool createHook preConfigurationHook --hookType PRE_CONFIGURATION --
procedureName <hookProcedure> --procedureProjectName <hookProject> --
resourceTemplateName <awsResourceTemplate> --projectName default
```

where `hookProcedure` is the your plugin name, `hookProject` is the name of the project to which the plugin belongs, and `awsResourceTemplate` is the name of the resource template that you created.

7. Create an environment template with the resource template that you created.

Example: Property Structure for a Cloud Provider Plugin

This is the EC-EC2 (Amazon) plugin property structure.

Property Name	Property Values
<code>ec_cloudprovisioning_plugin/</code>	
<code>displayName</code>	Amazon
<code>hasConfiguration</code>	1
<code>configurationLocation</code>	<code>ec2_cfgs</code>
<code>operations/</code>	<code>createConfiguration/</code> <code>deleteConfiguration/</code> <code>provision/</code> <code>retireResource/</code> <code>retireResourcePool/</code>
<code>ec_cloudprovisioning_plugin/createConfiguration/</code>	
<code>procedureName</code>	CreateConfiguration
<code>ui_formRefs/</code>	parameterForm called <code>ui_forms/EC2CreateConfigForm</code>

Property Name	Property Values
parameterRefs/	configuration called config
ec_cloudprovisioning_plugin/deleteConfiguration/	
procedureName	DeleteConfiguration
ui_formRefs/	–
parameterRefs/	configuration called config
ec_cloudprovisioning_plugin/provision/	
procedureName	API_RunInstances
ui_formRefs/	parameterForm called ec_parameterForm
parameterRefs/	configuration called config resourcePool called res_poolName count called count
ec_cloudprovisioning_plugin/retireResource/	
procedureName	API_TearDownResource
ui_formRefs/	–
parameterRefs/	resourcePool called res_poolName
ec_cloudprovisioning_plugin/retireResourcePool/	
procedureName	API_TearDownResource
ui_formRefs/	–
parameterRefs/	resourcePool called res_poolName

Creating Custom Configuration Management Plugins

This section applies to developers who want to create custom configuration management plugins for Dynamic Environments in ElectricFlow.

How to Create a Custom Configuration Management Plugin

1. To convert your procedure to a plugin, define the name of the plugin and the ElectricFlow project to which the plugin belongs.

Later in this procedure, you will use these properties to create pre- and post-hooks to the plugin in ElectricFlow.

2. Define the following properties in your plugin under a top-level plugin property called `ec_configurationmanagement_plugin`, which the Dynamic Environment system can access.

Name	Description
<code>ec_configurationmanagement_plugin/</code>	Top-level plugin property directory
Properties defined by the custom plugin under <code>ec_configurationmanagement_plugin</code>	
<code>displayName</code>	Name of the plugin that appears in the Dynamic Environment UI Example: Chef for the EC-Chef plugin
<code>hasConfiguration</code>	<Boolean flag - 0 1 true false> <ul style="list-style-type: none"> When this property is set to 0 or false, the plugin does not have any configuration procedures (CreateConfiguration and DeleteConfiguration). When this property is set to 1 or true, the plugin has configuration procedures.
<code>configurationLocation</code>	Name of the property sheet used by the plugin to store the saved configurations. This value is relative to the plugin's top-level properties. If this value is set as <i>chef_cfgs</i> , the configurations are in <i>/plugins/<PLUGIN_KEY>/project/chef_cfgs</i> .
<code>operations/</code>	Property sheet for the specific operations required by the Dynamic Environment system.
Properties under <code>operations/</code> – These operations are mapped to the plugin operations through the following child properties.	
<code>createConfiguration/</code>	Properties for the procedure that creates the plugin configuration. It is usually called <i>createConfiguration</i> .
<code>deleteConfiguration/</code>	Properties for the procedure that deletes the plugin configuration. It is usually called <i>deleteConfiguration</i> .

Name	Description
<code>converge/</code>	Properties for the procedure that converges the virtual instances to the defined configuration (including policies and roles).
<code>teardown/</code>	Properties for the procedure that deletes the configuration details on the specified dynamic resource or resource pool. For Chef, properties for the procedure that deletes the Chef API client and Chef node on the specified dynamic resource or resource pool.
Properties for the previously listed operations defined in the <code>operations</code> property sheet	Properties for the previously listed operations defined in the <code>operations</code> property sheet
<code>procedureName</code>	
<code>ui_formRefs/</code>	Property sheet that references the <code>ui_forms</code> properties defined under the plugin's top-level properties.
<code>parameterRefs/</code>	Property sheet for the input parameters to the procedure that can be used by the Dynamic Environment system. The parameter list is operation-specific.
Name	Description

Go to [Example: Property Structure for a Configuration Management Plugin](#) on page 301 for an example of the properties for the EC-Chef plugin.

3. Save the plugin file in the appropriate location on your ElectricFlow server.
4. To import the plugin file to the automation platform:
 - a. In the automation platform, go to **Administration > Plugins** to open the **Plugin Manager** page.
 - b. Click the **Install from File/URL** tab.
 - c. In the **File Install** field, click **Choose file** to select the plugin file.
 - d. Click **Upload** to install it.

The plugin file appears in the **Currently Installed** tab.
 - e. Find your plugin and click **Promote** in the Actions column to make it available for use by ElectricFlow.

5. In ElectricFlow, to create a resource template:

- a. Go to [Creating Resource Templates](#) on page 251 for the details.
- b. In the **Select cloud provider** field, select your plugin as the cloud provider.
- c. Enter the plugin settings in the form.
- d. Set the configuration management tool.

Go to [Creating Resource Templates](#) on page 251 for the details.

6. In the ectool API, enter the following commands to set pre- and post-hooks linking the plugin to an application that you later deploy in a dynamic environment.

- a. To create a pre-hook, enter

```
ectool createHook postConfigurationHook --hookType POST_CONFIGURATION --
procedureName <hookProcedure> --procedureProjectName <hookProject> --
resourceTemplateName <awsResourceTemplate> --projectName default
```

where `hookProcedure` is the your plugin name, `hookProject` is the name of the project to which the plugin belongs, and `awsResourceTemplate` is the name of the resource template that you created.

- b. To create a post-hook, enter

```
ectool createHook preConfigurationHook --hookType PRE_CONFIGURATION --
procedureName <hookProcedure> --procedureProjectName <hookProject> --
resourceTemplateName <awsResourceTemplate> --projectName default
```

where `hookProcedure` is the your plugin name, `hookProject` is the name of the project to which the plugin belongs, and `awsResourceTemplate` is the name of the resource template that you created.

7. Create an environment template with the resource template that you created.

Example: Property Structure for a Configuration Management Plugin

This is the EC-Chef plugin property structure.

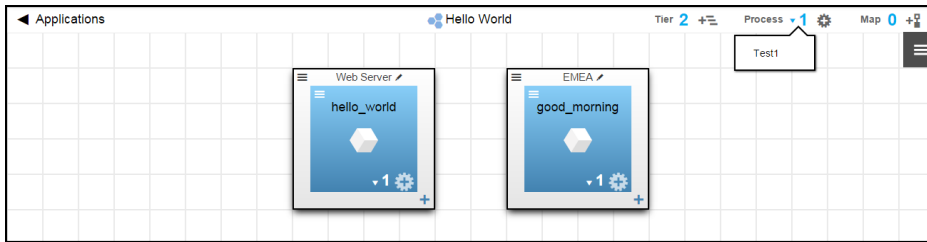
Name	Values
ec_configurationmanagement_plugin/	
displayName	Chef
hasConfiguration	1
configurationLocation	chef_cfgs

Name	Values
operations/	createConfiguration/ deleteConfiguration/ converge/ teardown/
ec_configurationmanagement_plugin/createConfiguration/	
procedureName	CreateConfiguration
ui_formRefs/	parameterForm called forms/CreateConfigForm
parameterRefs/	configuration called config
ec_configurationmanagement_plugin/deleteConfiguration/	
procedureName	DeleteConfiguration
ui_formRefs/	–
parameterRefs/	configuration called config
ec_configurationmanagement_plugin/converge/	
procedureName	_RegisterAndConvergeNode
ui_formRefs/	parameterForm called ec_parameterForm
parameterRefs/	configuration called config
ec_cloudprovisioning_plugin/teardown/	
procedureName	_DeleteNode
ui_formRefs/	–
parameterRefs/	resourceName called resource_name

Deploying Applications With Provisioned Cloud Resources

About the example in this topic:

The example in this topic consists of an application called "Hello World" application with one application process called *Test1*, which has two steps.



Test1 will be deployed in a dynamic environment called *CloudEnv*. You model the CloudEnv environment using the *AWSIest* environment template.



The AWSIest environment template has three tiers:

- Tier 1 and Tier 3 have cloud resources defined in resource templates. These resources can be provisioned when you deploy the application.
- Tier 2 has static resources, which cannot be provisioned.

Starting in the Home page:

1. Go to the Applications List.
 - Starting from the Main menu, click the **Menu** button, and then select **Applications**.
 - Starting from the Home page, click **Applications**.

The Applications List opens.

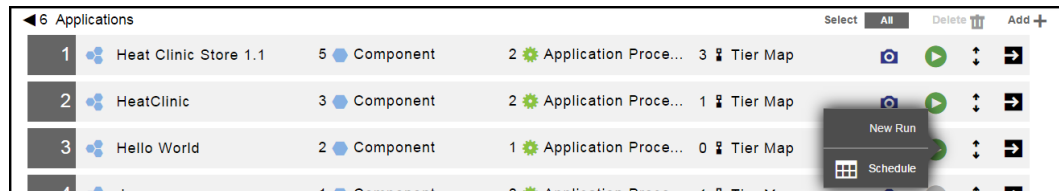
Example:

This example uses the "Hello World" application.

6 Applications					Select	All	Delete	Add
1	Heat Clinic Store 1.1	5 Component	2 Application Process	3 Tier Map				
2	HeatClinic	3 Component	2 Application Process	1 Tier Map				
3	Hello World	2 Component	1 Application Process	0 Tier Map				

2. Choose an application, and click the **Run process** button for that application.

Example:

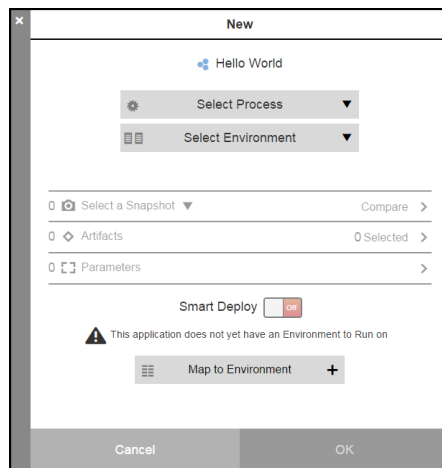


3. Click **New Run**.

The **New** dialog box to deploy the application opens.

See the messages in this dialog box for hints about what you need to do to deploy the application.

Example:

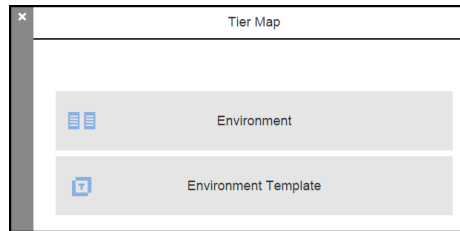


4. Select the application process to deploy.

- Click **Map to Environment +** to create a tier map for the application.

The **Tier Map** dialog box opens.

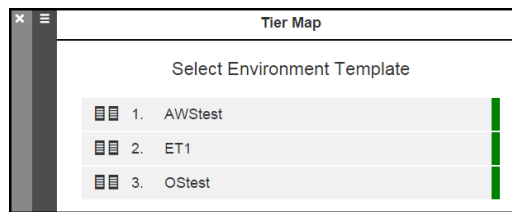
Example:



- Select **Environment Template**.

The **Tier Map** dialog box to select an environment tier opens.

Example:

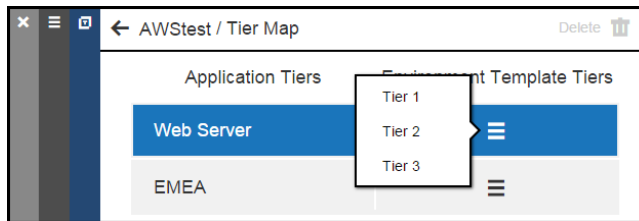


- Select an environment template.

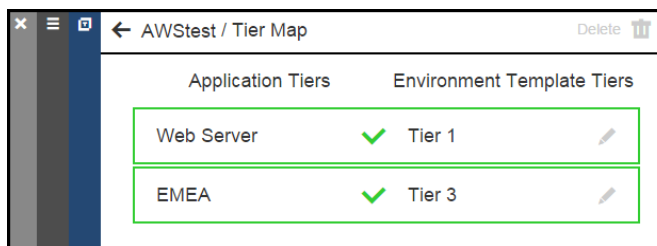
The <Environment template>/Tier Map dialog box opens.

- For each application tier, click the **Menu** button to select an environment tier to which the application tier is mapped.

Example:



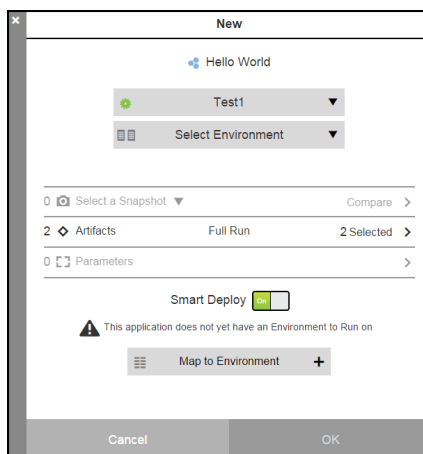
Example:



- Click **OK**.

The **New** dialog box to deploy the application re-opens.

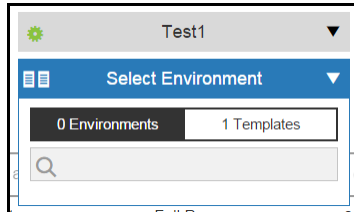
Example:



10. Click **Select Environment** to select an environment template.

A list of available environments and environment template based on the tier map opens.

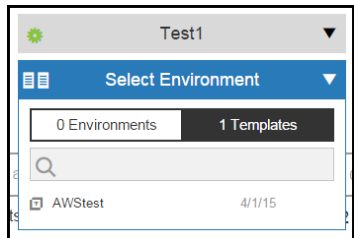
Example:



11. Select **1 Templates**.

A list of available environment templates appears.

Example:

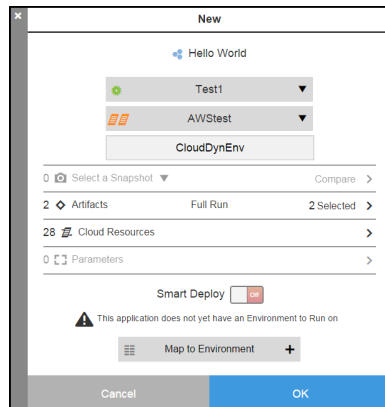


12. Click the environment template that you want to use.

13. Enter a name for the dynamic environment that will be created from the selected environment template.

The **New** dialog box to deploy the application now shows the environment template name below the application process name. It also shows the number of cloud resources provisioned in the environment templates.

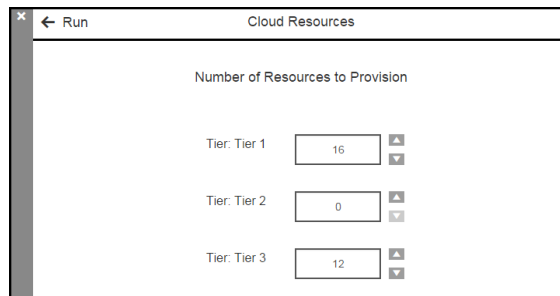
Example:



14. Click in the **Cloud Resources** row.

The **Cloud Resources** dialog box opens.

Example:



15. Change the number of cloud resources to provision

Example:

In this example, Tier 1 and Tier 3 have one or more resources to provision because they have cloud resources. You cannot provision resources in Tier 2 because it has only static resources.

16. Click **OK**.

The **New** dialog box now shows the new number of resources to provision.

The message "This application does not yet have an Environment to Run on" still appears.

When you click **OK**, ElectricFlow first attempts to create the dynamic environment. If this is successful, it deploys the application.

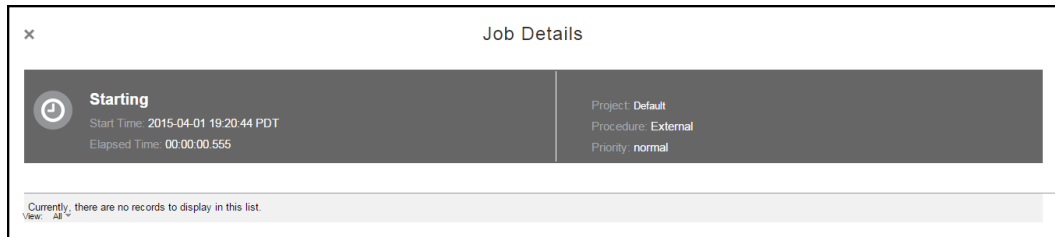
Example:

The Applications List opens.

- To view the job details, click the **View Details** button to open to the Job Details page.

ElectricFlow first runs the job to create the dynamic environment. If this job is successful, it deploys the application.

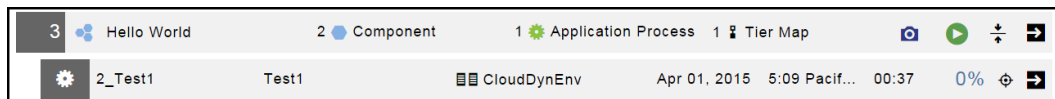
Example:



You can also see the status of jobs as they progress in the Applications List.

Example:

This example shows the Application List when the job starts.



If you provision resources that are not available, such as 20 cloud resources in Tier 1 and 3 static resources in Tier 2, the job to create the dynamic environment fails and the application is not deployed.

Deploying Applications with Parameters

Starting in the Applications List:

- Choose the application that you want to deploy and click the **Run process** button.
- Select **New Run** to deploy the application with new settings.

The **New** dialog box opens.

- Select the application process, environment, tier map, snapshot, artifact, and resource options, as described in other topics in this document.



- In the **Parameters** row, click the button to open a form showing the parameters that apply to the application process.
- Enter information in the fields.
- Click **OK** to save the parameter settings and close the form.

The **New** dialog box now shows what you set in the previous two steps.

The Parameters row shows the number of required parameters. You must enter information for those

parameters to deploy the application.

7. Click **OK** to deploy the application.

Retiring Dynamic Environments

After you have deployed your application in a dynamic environment, you can retire it and destroy the resources, which makes them unavailable for any future deployments.

Starting in the Environments List:

1. Choose a dynamic environment.

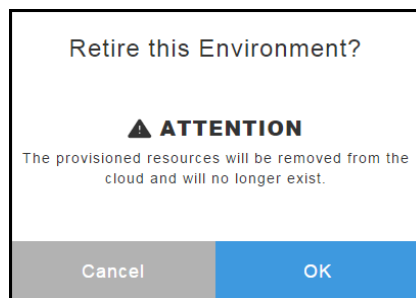
Example:

10 Environments					Lists		Select	Delete	Add
10 Environments 3 Environment Templates 3 Resource Templates									
1	CloudDynEnv	2 Tiers	0 Resources	0 Applications installed	Stop	Refresh	More	Up	Down
2	CloudEnv	2 Tiers	3 Resources	0 Applications installed	Stop	Refresh	More	Up	Down
3	hc-store dev	3 Tiers	3 Resources	0 Applications installed	Stop	Refresh	More	Up	Down

2. Click the **Tear down** button for that environment.

A message appears.

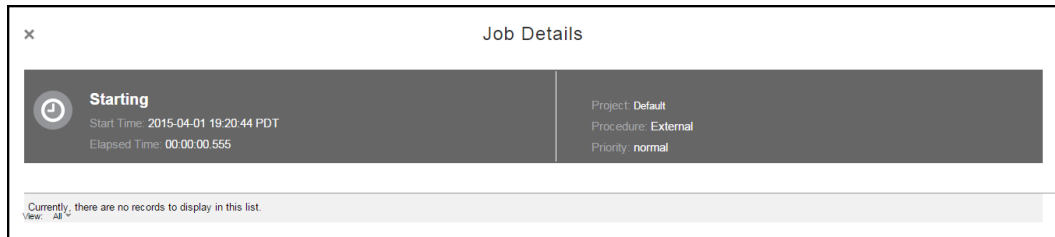
Example:



- Click **OK** to verify that you want to retire the environment.

The Job Details page opens.

Example:



- View the results in the Job Details page.

Dynamic Environment Example with Amazon and Chef

This example shows what you see in the ElectricFlow UI when your dynamic environment is configured with Amazon as the cloud provider and Chef for the configuration management.

Resource Templates

In resource templates, you define cloud resources that will be provisioned for dynamic environments. Examples of cloud resources are a HAProxy server, MySQL server, PHP-FPM server, and Resque server.

This is the Resource Template List.

0 Environments		Lists				Select	Alt	Delete	Add
8 Environments		0 Environment Templates		4 Resource Templates					
1	HAProxy_Image	0 Pools	Amazon	Created: 03/31/15	By: admin				
2	MySQL_Image	0 Pools	Amazon	Created: 03/31/15	By: admin				
3	PHP-FPM_Image	0 Pools	Amazon	Created: 03/31/15	By: admin				
4	Resque_Image	0 Pools	Amazon	Created: 03/31/15	By: admin				

These are the cloud provider settings for a resource template. It is defined by cloud provider account credentials and by an Amazon Machine Image (AMI).

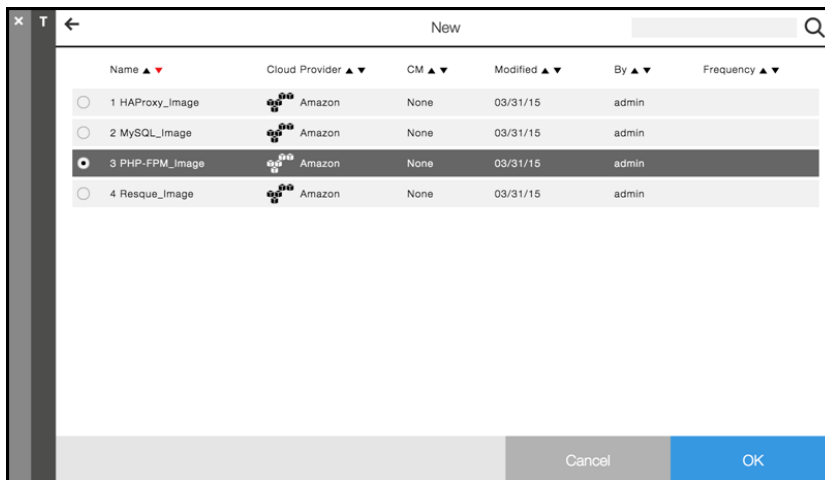
The screenshot shows a web interface for creating a new resource template. The title bar says 'New'. Below it, the 'Resource Template Details' section has two tabs: 'Provider' and 'Configuration Management'. The 'Account Setup' section shows 'Amazon' selected in a dropdown menu, with 'EC AWS 1' selected in a sub-dropdown. Below this, there are five text input fields: 'Configuration Name' (EC AWS 1), 'Description' (EC2 integration), 'Service URL' (https://ec2.amazonaws.com), 'Resource Pool' (cloud_servers), and 'Workspace' (default). At the bottom right are 'Cancel' and 'Next' buttons.

These are the configuration management settings for a resource template. It calls a Chef recipe to configure the cloud resources. You can configure the resources with other recipes or predefined configuration, such as a MySQL stack configuration or a HAProxy load balancer configuration.

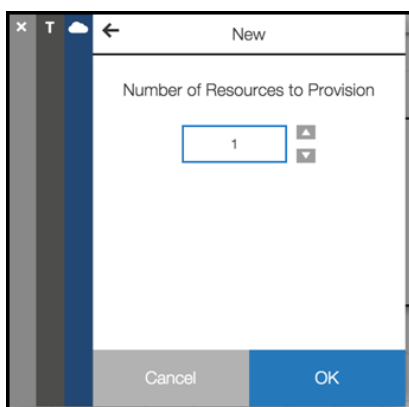
The screenshot shows the same web interface but with 'Chef' selected in the 'Account Setup' dropdown menu. The sub-dropdown shows 'Select account'. The 'Configuration Name' field contains 'EC-SJ-ChefServer', the 'Description' field contains 'Chef configuration', and the 'Chef Server URL' field contains 'https://ce.chefserver.internal.com/123'. There is also a 'I run as:' label with a user icon. The 'Cancel' and 'Next' buttons are at the bottom right.

Environment Templates

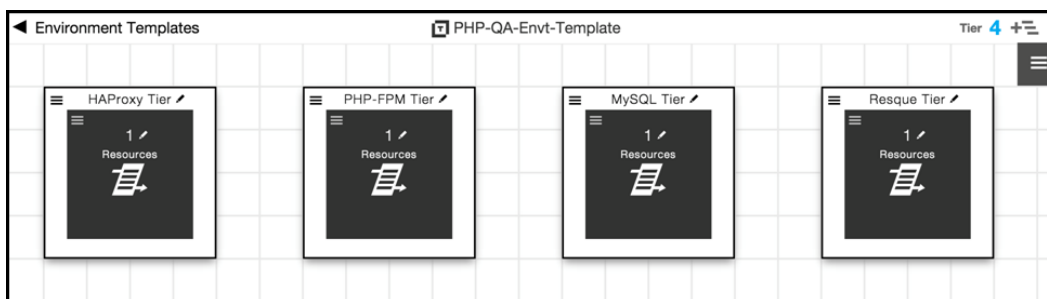
When you create a new environment template, you can add one resource template to an environment tier. These are the resource templates from which you can select.



After you select a resource template and click **OK**, you enter the number of resources to provision in an environment tier.

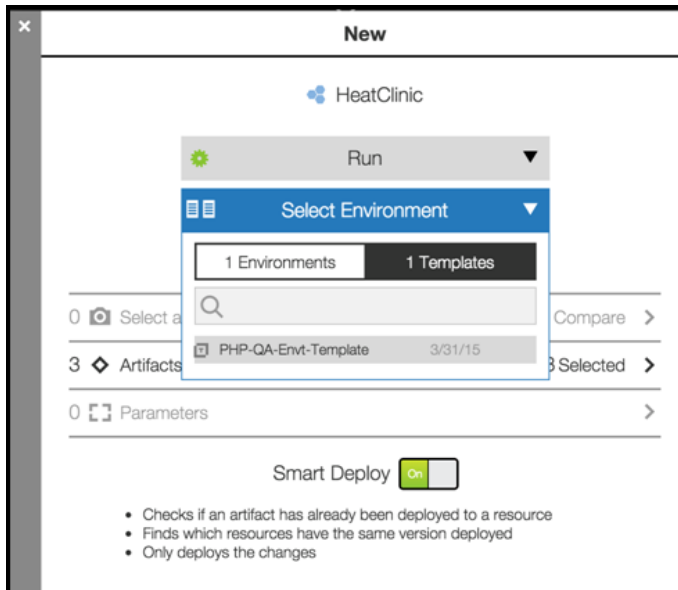


This is an environment template with four tiers. Each tier is configured with a resource template, and resource template has a provisioned resource.



Deploying Applications to Dynamic Environments

After you select an application to deploy, you select the application process to run and the environment in which to deploy the application.



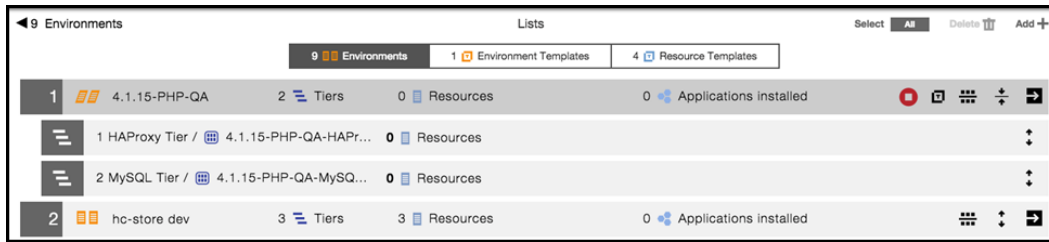
To deploy the application in a dynamic environment, you can select an environment template. A dynamic environment is created with the provisioned cloud resources. After the resources are provisioned and the dynamic environment is created, the application is deployed.

You can view the status of the provisioning process.

Applications / View Run						admin Running: HeatClinic - Run on 4.1.15-PHP-...		Errors 0	
10_Run_HeatClinic_Default_201503311031...	Mar 31, 2015	10:31 PDT	00:19	0%					
Provisioning - 4.1.15-PHP-QA - HAProxy...	Mar 31, 2015	10:31 PDT	00:14	25%					
Provisioning - 4.1.15-PHP-QA - MySQL...	Mar 31, 2015	10:31 PDT	00:14	25%					
Handle Provisioning and Configuring Err...	Mar 31, 2015	10:31 PDT	00:00	0%					

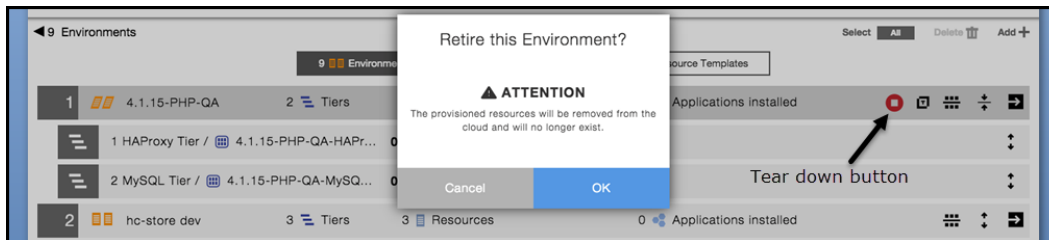
Retiring Dynamic Environments

After the provisioning process completes, the new dynamic environment appears in the Environments List.



You can retire dynamic environments on an on-demand basis to prevent excessive use of cloud resources and reduce costs.

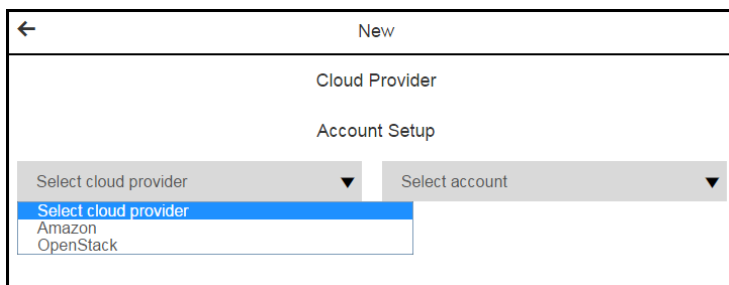
You click the **Tear down** button to retire the selected dynamic environment.



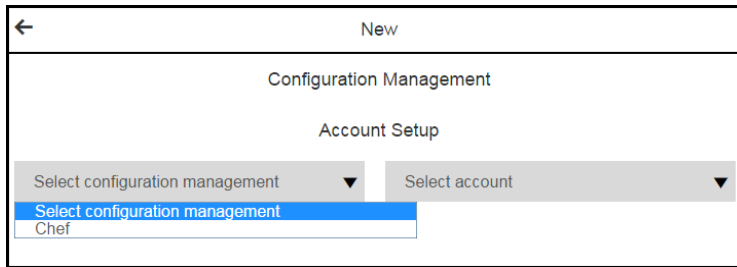
Developer Task: Creating Custom Plugins

ElectricFlow uses third-party plugins to provision cloud resources in resource templates for Dynamic Environments. Amazon EC2 (EC-EC2) and Openstack (EC-OpenStack) are supported as out-of-the-box cloud provider plugins, and Chef (EC-Chef) is supported as an out-of-the-box configuration management plugin.

To view the supported cloud provider plugins, open the **New Cloud Provider** dialog box, and click **Select cloud provider** to view the drop-down menu of cloud providers.



To view the supported configuration management plugins, click **Select configuration management** to view the drop-down menu of configuration management tools.



The screenshot shows a mobile application interface for setting up configuration management. At the top is a 'New' header with a back arrow. Below it is a 'Configuration Management' section. Under this section is an 'Account Setup' header. There are two dropdown menus: 'Select configuration management' and 'Select account'. The 'Select configuration management' dropdown is open, showing a list with 'Select configuration management' at the top and 'Chef' below it. The 'Select account' dropdown is closed.

If you have an existing procedure that you would prefer to use instead of an out-of-the-box third-party plugin to provision cloud resources, you can create a custom plugin based on your deployment scenario.

The process to create custom cloud provider and configuration management plugins uses metadata to loosely couple them to the application you want to deploy. The custom plugins define specific known properties that are automatically recognized by Dynamic Environments.

To create your own plugin for other cloud platforms, go to <https://electric-cloud.github.io/index.html> to get started. This website describes how to create open source plugins and has this information:

- For an introduction to the process of building and developing open source plugins, go to the [Quick Start](#).
- To build and deploy your plugin, go to [Build and Deploy](#).
- For more information, go to [More information](#).
- Go to [Creating Custom Cloud Provider Plugins](#) on page 317 for the details to create cloud provider plugins.
- Go to [Creating Custom Configuration Management Plugins](#) on page 321 for the details to create configuration management plugins.

Creating Custom Cloud Provider Plugins

This section applies to developers who want to create custom cloud provider plugins for Dynamic Environments in ElectricFlow.

How to Create a Custom Cloud Provider Plugin

1. To convert your procedure to a plugin, define the name of the plugin and the ElectricFlow project to which the plugin belongs.

Later in this procedure, you will use these properties to create pre- and post-hooks to the plugin in ElectricFlow.

2. Define the following properties in your plugin under a top-level plugin property called `ec_cloudprovisioning_plugin`, which the Dynamic Environment system can access.

Property Name	Description
<code>ec_cloudprovisioning_plugin/</code>	Top-level plugin property directory
Properties defined by the custom plugin under <code>ec_cloudprovisioning_plugin</code>	
<code>displayName</code>	<p>Name of the plugin that appears in the New Cloud Provider dialog box.</p> <p>Example: Amazon for the EC-EC2 plugin</p>
<code>hasConfiguration</code>	<p><Boolean flag - 0 1 true false></p> <ul style="list-style-type: none"> When this property is set to 0, the plugin does not have any configuration procedures (<code>CreateConfiguration</code> and <code>DeleteConfiguration</code>). When this property is set to 1, the plugin has configuration procedures.
<code>configurationLocation</code>	<p>Name of the property sheet used by the plugin to store the saved configurations.</p> <p>This value is relative to the plugin's top-level properties.</p> <p>If this value is set as <code>ec2_cfgs</code>, the configurations are in <code>/plugins/<PLUGIN_KEY>/project/ec2_cfgs</code>.</p>
<code>operations/</code>	Property sheet for the specific operations required by the Dynamic Environment system.
Properties under <code>operations/</code> – These operations are mapped to the plugin operations through the following child properties.	
<code>createConfiguration/</code>	Properties for the procedure that creates the plugin configuration. It is usually called <code>createConfiguration</code> .
<code>deleteConfiguration/</code>	Properties for the procedure that deletes the plugin configuration. It is usually called <code>deleteConfiguration</code> .
<code>provision/</code>	Properties for the procedure that provisions virtual instances.
<code>retireResource/</code>	Properties for the procedure that tears down a previously provisioned virtual instance backing the specified resource.

Property Name	Description
<code>retireResourcePool/</code>	Properties for the procedure that tears down all previously provisioned virtual instances for the specified resource pool.
Properties for the previously listed operations defined in the <code>operations</code> property sheet	
<code>procedureName</code>	Name of the procedure name in the plugin to which the operation is mapped.
<code>ui_formRefs/</code>	Property sheet that references the <code>ui_forms</code> properties defined under the plugin's top-level properties.
<code>parameterRefs/</code>	Property sheet for the input parameters to the procedure that can be used by the Dynamic Environment system. The parameter list is operation-specific.

Go to [Example: Property Structure for a Cloud Provider Plugin](#) on page 320 for an example of the properties for the EC-EC2 (Amazon) plugin.

3. Save the plugin file in the appropriate location on your ElectricFlow server.
4. To import the plugin file to the automation platform:
 - a. In the automation platform, go to **Administration > Plugins** to open the **Plugin Manager** page.
 - b. Click the **Install from File/URL** tab.
 - c. In the **File Install** field, click **Choose file** to select the plugin file.
 - d. Click **Upload** to install it.

The plugin file appears in the **Currently Installed** tab.
 - e. Find your plugin and click **Promote** in the Actions column to make it available for use by ElectricFlow.
5. In ElectricFlow, to create a resource template:
 - a. Go to [Creating Resource Templates](#) on page 251 for the details.
 - b. In the **Select cloud provider** field, select your plugin as the cloud provider.
 - c. Enter the plugin settings in the form.
 - d. (Optional) Set the configuration management tool.

Go to [Creating Resource Templates](#) on page 251 for the details.

6. In the ectool API, enter the following commands to set pre- and post-hooks linking the plugin to an application that you later deploy in a dynamic environment.

- a. To create a pre-hook, enter

```
ectool createHook postConfigurationHook --hookType POST_CONFIGURATION --
procedureName <hookProcedure> --procedureProjectName <hookProject> --
resourceTemplateName <awsResourceTemplate> --projectName default
```

where `hookProcedure` is the your plugin name, `hookProject` is the name of the project to which the plugin belongs, and `awsResourceTemplate` is the name of the resource template that you created.

- b. To create a post-hook, enter

```
ectool createHook preConfigurationHook --hookType PRE_CONFIGURATION --
procedureName <hookProcedure> --procedureProjectName <hookProject> --
resourceTemplateName <awsResourceTemplate> --projectName default
```

where `hookProcedure` is the your plugin name, `hookProject` is the name of the project to which the plugin belongs, and `awsResourceTemplate` is the name of the resource template that you created.

7. Create an environment template with the resource template that you created.

Example: Property Structure for a Cloud Provider Plugin

This is the EC-EC2 (Amazon) plugin property structure.

Property Name	Property Values
ec_cloudprovisioning_plugin/	
displayName	Amazon
hasConfiguration	1
configurationLocation	ec2_cfgs
operations/	createConfiguration/ deleteConfiguration/ provision/ retireResource/ retireResourcePool/
ec_cloudprovisioning_plugin/createConfiguration/	
procedureName	CreateConfiguration
ui_formRefs/	parameterForm called ui_forms/EC2CreateConfigForm

Property Name	Property Values
parameterRefs/	configuration called config
ec_cloudprovisioning_plugin/deleteConfiguration/	
procedureName	DeleteConfiguration
ui_formRefs/	–
parameterRefs/	configuration called config
ec_cloudprovisioning_plugin/provision/	
procedureName	API_RunInstances
ui_formRefs/	parameterForm called ec_parameterForm
parameterRefs/	configuration called config resourcePool called res_poolName count called count
ec_cloudprovisioning_plugin/retireResource/	
procedureName	API_TearDownResource
ui_formRefs/	–
parameterRefs/	resourcePool called res_poolName
ec_cloudprovisioning_plugin/retireResourcePool/	
procedureName	API_TearDownResource
ui_formRefs/	–
parameterRefs/	resourcePool called res_poolName

Creating Custom Configuration Management Plugins

This section applies to developers who want to create custom configuration management plugins for Dynamic Environments in ElectricFlow.

How to Create a Custom Configuration Management Plugin

1. To convert your procedure to a plugin, define the name of the plugin and the ElectricFlow project to which the plugin belongs.

Later in this procedure, you will use these properties to create pre- and post-hooks to the plugin in ElectricFlow.

2. Define the following properties in your plugin under a top-level plugin property called `ec_configurationmanagement_plugin`, which the Dynamic Environment system can access.

Name	Description
<code>ec_configurationmanagement_plugin/</code>	Top-level plugin property directory
Properties defined by the custom plugin under <code>ec_configurationmanagement_plugin</code>	
<code>displayName</code>	Name of the plugin that appears in the Dynamic Environment UI Example: Chef for the EC-Chef plugin
<code>hasConfiguration</code>	<Boolean flag - 0 1 true false> <ul style="list-style-type: none"> When this property is set to 0 or false, the plugin does not have any configuration procedures (CreateConfiguration and DeleteConfiguration). When this property is set to 1 or true, the plugin has configuration procedures.
<code>configurationLocation</code>	Name of the property sheet used by the plugin to store the saved configurations. This value is relative to the plugin's top-level properties. If this value is set as <i>chef_cfgs</i> , the configurations are in <i>/plugins/<PLUGIN_KEY>/project/chef_cfgs</i> .
<code>operations/</code>	Property sheet for the specific operations required by the Dynamic Environment system.
Properties under <code>operations/</code> – These operations are mapped to the plugin operations through the following child properties.	
<code>createConfiguration/</code>	Properties for the procedure that creates the plugin configuration. It is usually called <i>createConfiguration</i> .
<code>deleteConfiguration/</code>	Properties for the procedure that deletes the plugin configuration. It is usually called <i>deleteConfiguration</i> .

Name	Description
<code>converge/</code>	Properties for the procedure that converges the virtual instances to the defined configuration (including policies and roles).
<code>teardown/</code>	Properties for the procedure that deletes the configuration details on the specified dynamic resource or resource pool. For Chef, properties for the procedure that deletes the Chef API client and Chef node on the specified dynamic resource or resource pool.
Properties for the previously listed operations defined in the <code>operations</code> property sheet	Properties for the previously listed operations defined in the <code>operations</code> property sheet
<code>procedureName</code>	
<code>ui_formRefs/</code>	Property sheet that references the <code>ui_forms</code> properties defined under the plugin's top-level properties.
<code>parameterRefs/</code>	Property sheet for the input parameters to the procedure that can be used by the Dynamic Environment system. The parameter list is operation-specific.
Name	Description

Go to [Example: Property Structure for a Configuration Management Plugin](#) on page 325 for an example of the properties for the EC-Chef plugin.

3. Save the plugin file in the appropriate location on your ElectricFlow server.
4. To import the plugin file to the automation platform:
 - a. In the automation platform, go to **Administration > Plugins** to open the **Plugin Manager** page.
 - b. Click the **Install from File/URL** tab.
 - c. In the **File Install** field, click **Choose file** to select the plugin file.
 - d. Click **Upload** to install it.

The plugin file appears in the **Currently Installed** tab.
 - e. Find your plugin and click **Promote** in the Actions column to make it available for use by ElectricFlow.

5. In ElectricFlow, to create a resource template:

- a. Go to [Creating Resource Templates](#) on page 251 for the details.
- b. In the **Select cloud provider** field, select your plugin as the cloud provider.
- c. Enter the plugin settings in the form.
- d. Set the configuration management tool.

Go to [Creating Resource Templates](#) on page 251 for the details.

6. In the ectool API, enter the following commands to set pre- and post-hooks linking the plugin to an application that you later deploy in a dynamic environment.

- a. To create a pre-hook, enter

```
ectool createHook postConfigurationHook --hookType POST_CONFIGURATION --
procedureName <hookProcedure> --procedureProjectName <hookProject> --
resourceTemplateName <awsResourceTemplate> --projectName default
```

where `hookProcedure` is the your plugin name, `hookProject` is the name of the project to which the plugin belongs, and `awsResourceTemplate` is the name of the resource template that you created.

- b. To create a post-hook, enter

```
ectool createHook preConfigurationHook --hookType PRE_CONFIGURATION --
procedureName <hookProcedure> --procedureProjectName <hookProject> --
resourceTemplateName <awsResourceTemplate> --projectName default
```

where `hookProcedure` is the your plugin name, `hookProject` is the name of the project to which the plugin belongs, and `awsResourceTemplate` is the name of the resource template that you created.

7. Create an environment template with the resource template that you created.

Example: Property Structure for a Configuration Management Plugin

This is the EC-Chef plugin property structure.

Name	Values
ec_configurationmanagement_plugin/	
displayName	Chef
hasConfiguration	1
configurationLocation	chef_cfgs

Name	Values
operations/	createConfiguration/ deleteConfiguration/ converge/ teardown/
ec_configurationmanagement_plugin/createConfiguration/	
procedureName	CreateConfiguration
ui_formRefs/	parameterForm called forms/CreateConfigForm
parameterRefs/	configuration called config
ec_configurationmanagement_plugin/deleteConfiguration/	
procedureName	DeleteConfiguration
ui_formRefs/	–
parameterRefs/	configuration called config
ec_configurationmanagement_plugin/converge/	
procedureName	_RegisterAndConvergeNode
ui_formRefs/	parameterForm called ec_parameterForm
parameterRefs/	configuration called config
ec_cloudprovisioning_plugin/teardown/	
procedureName	_DeleteNode
ui_formRefs/	–
parameterRefs/	resourceName called resource_name

Master Components

The *master component* feature promotes component reuse. It provides an easy way to create new components by re-using existing component definitions and the underlying process definitions. Master components are project-scoped components that can be created as new objects or from an existing master or application component. A master component consists of a component definition and one or more component processes.

You can use master components to create standardized reusable components and to promote best practices across the enterprise. When deployments have large numbers of similar components, this can be used to speed up application authoring by leveraging prebuilt component and component-process logic. One master component can include processes used in different parts of one or more applications. You can model these applications using the same master component and deploy them more than once.

In this release, you can create reusable components at the project level and later use them to quickly create application components when modeling applications. You copy a master component or an existing application component to create a new component in an application. The new component will have the same properties as the master or existing application component. Any changes that you make after this do not affect the master or existing application component. You can also save existing components as master components that other users can reuse and share.

After an application component is created from a master component, any subsequent changes to the master component will not affect the application component. To get the latest changes in the application component, delete and recreate the application component by copying the master component.

Creating a New Master Component

Starting in the Master Components List or the Applications List:

1. Click the **Add +** button.
The New dialog box opens.
2. Click **Create new Master Component**.
3. Enter the name of the new master component.
4. (Optional) Enter the description of the new component.
5. Click the down arrow in the **Content Location** field.
6. Select a plugin.
7. Enter the information in the fields for the plugin.
8. Click **OK**.

The component that you copied now appears in the list.

Next steps:

- Add or edit an component process to the component.
- Create an application component in an application.

Creating a Master Component Based on an Existing Master Component

Starting in the Master Components List or the Applications List:

1. Click the **Add +** button.
The **New** dialog box opens.
2. Click **Create new Master Component**.
3. Click **Create from existing master component**.
4. Select a component in the list.

5. Change the name of the component.
6. (Optional) Change the description of the component.
7. Click **OK**.

The component that you copied now appears in the list.

Next steps:

- Add or edit an component process to the component.
- Create an application component in an application.

Creating a Master Component Based on an Existing Application Component

Starting in the Master Components List or the Applications List:

1. Click the **Add +** button.
The New dialog box opens.
2. Click **Create new Master Component**.
3. Click **Create from existing application component**.
4. Select an application.

A list of components for the application appear.

5. (Optional) Select an application tier to refine the list of components.
6. Select a component.
7. Change the name of the component.
8. (Optional) Change the description of the component.
9. Click **OK**.

The component that you copied now appears in the list.

Next steps:

- Add or edit an component process to the component.
- Create an application component in an application.

Adding a Component Process to a Master Component

Starting In the Master Components List:

1. Choose a master component.
2. Click the **Menu** button.
3. Select **Add process**.

The Component Process Details dialog box opens.

4. Enter a process name, and continue with the normal authoring process.
5. Click **OK**.

Next steps:

- Model the component process in the Component Process visual editor.
- Create an application component in an application.

Editing a Master Component

Starting In the Master Components List:

1. Click the **Menu** button for a master component.
2. Select **Details**.
The Component Details dialog box opens.
3. Change one or more settings.
4. Click **OK**.

Next steps:

- Add an component process to the component.
- Create an application component in an application.

Creating an Application Component Based on a Master Component

Starting in the Applications List:

1. Select an application.
2. Go to the Application Visual Editor for the selected application.
3. Click **+** in an application tier to add a component to it.
4. Click **+** in the component.
5. Click **Create from existing master component**.
6. Select an existing master component in the list.
7. Rename the component.
8. (Optional) Edit the description of the component.
9. Click **OK**.

Next steps:

- View the component processes for this component. Note that the component has the same component processes as the master component.
- Edit the details or a component process for this component.
- Add a new component process to this component.

Creating an Application Component Based on an Existing Application Component

Starting in the Applications List:

1. Select an application.
2. Go to the Application Visual Editor for the selected application.
3. Click **+** in an application tier to add a component to it.
4. Click **+** in the component.
5. Click **Create from existing application component**.
6. Select an application. A list of components for the application appear.
7. (Optional) Select an application tier to refine the list of components.
8. Select a component.
9. Change the name of the component.
10. (Optional) Change the description of the component.
11. Click **OK**.

The component that you copied now appears in the list.

Next steps:

- View the component processes for this component.
Note that the component has the same component processes as the master component.
- Edit the details or a component process for this component.
- Add a new component process to this component.

Deleting Master Components

Starting in the Master Component List:

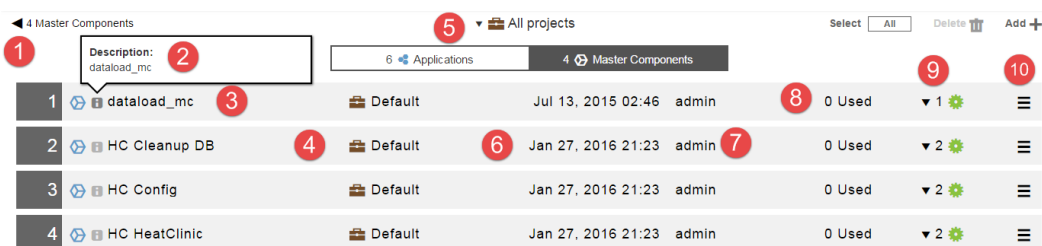
1. Select one or more master components.
To select all or none of the items in the list, click **Select** and toggle between **All** or **None**.
2. Click the **Delete** button.
An message appears asking you to verify that you want to delete the component.
3. Click **OK** to delete the selected master components.
The Master Components List is updated.

Master Components List UI

How to get here:

One of these ways:

- From the Home page:
 1. Click the **Applications** launch pad to open the Applications List.
 2. Click the **Master components** tab.
- Click the **Main menu** button, and then click **Applications > Master Components**.



1	Breadcrumb showing the total number of applications in your ElectricFlow system.
2	You can view a list of the applications or the master components in your system. Select the Master components tab to view the list of applications.
3	Click the Select button to select all or none of the items in the list. Click the Delete button to delete the selected items. Click the Add + button to add a master component.
4	Name of the master component.
5	When the master component was created.
6	The user who created this component.
7	Number of component processes that have the component in a component process step. Click the down arrow to add a new component process for the master component.
8	Click the Menu button to open the context menu. You can get more details about the master component: <ul style="list-style-type: none"> • Details—The name and description of the object. • Add Process—Add a component process. The Component Process Details dialog box opens. A master component can have one or more component processes. • Properties—The properties in the object. • Access Control—The access control configuration in the automation platform for the object. • Track Changes—The change history of the object. • Delete—Delete this object.

Deploying Applications in ElectricFlow

How you deploy applications in ElectricFlow depends on what you want to deploy:

- Full deploy—ElectricFlow deploys all the objects (including application processes, components, and artifacts) in the application. When you deploy an application for the first time, you must do a full deploy. Later, you can use this method to verify that the software is ready to be released.
- Smart deploy—The system deploys only the artifacts that have not been deployed to a resource, specific versions of the artifacts, or artifacts that have not yet been deployed to new resources. You can use this method during the development and test phases to verify artifact or resource updates.
- Partial deploy—The system deploys only objects that you select. You can use this method during the development and test phases to verify incremental changes.
- Partial deploy with specific artifact versions—The system deploys only the artifacts with selected versions. You can use this method during the development and test phases to verify incremental changes.
- Schedule—You can create schedules to deploy applications on a one-time, daily, weekly, or monthly basis. You can schedule nightly builds so that the developers always have a new version to continue their work on every day.
- Snapshot—You select a snapshot to deploy. A snapshot is an immutable version of the application with specific artifact versions. When you save snapshots during the build, test, deploy, and release phases, you can compare them to optimize and troubleshoot the application in ElectricFlow.

Note: If you select a snapshot and modify it, it is no longer a snapshot and becomes a different version of the application, which you can save as a new snapshot.

- Based on custom parameters defined in application processes—Set these parameters when you deploy the application or when you define an application process step. They determine how the application should be deployed.

You can run applications by combining methods, such as smart deploy and a partial deploy with specific artifact versions, full deploy with a snapshot, or only smart deploy. This sequence is an example of how you may deploy your software over time.

- When you create an application in ElectricFlow and deploy it for the first time, you must do a full deploy. By default, smart deploy is disabled the first time that you run an application.
- Throughout these deployments, you can schedule the deployments to occur on a daily or weekly basis to provide builds that can be tested and installed on a regular basis.
- You can also save snapshots of the software at regular intervals or milestones. When you need to troubleshoot the software or want to do performance testing, you can use one of these snapshots for comparison.
- If the application does not deploy successfully, you can redeploy parts of it to troubleshoot the application or component processes that failed.

You can do a partial deploy and redeploy the application with only the objects that failed.

You can also do a partial deploy only with specific versions of artifacts to determine if one or more specific versions of artifacts are causing problems.

- Later, after successfully deploying the application, you can redeploy parts of the application when new versions of artifacts or new resources are available.

- When a new version of an artifact is released, you can deploy only the artifact by selecting the new version and doing a partial deploy.
- When you add artifacts and resources to the application, you deploy the new artifacts to resources and specific versions of selected artifacts to the new resources, a combination of smart deploy and partial deploy with specific artifact versions.


Deployment Automation User Interface




This section describes the ElectricFlow user interface (UI) for deployment automation.








ElectricFlow Buttons and Icons





These buttons and icons appear in the ElectricFlow user interface (UI).






For more information about how they work in the UI, see [Deployment Automation User Interface](#) on page 333.



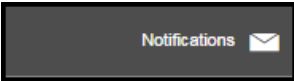


Button or Icon	Name	Tooltip (Hover over the button to view the tooltip.)	How to Use It
	Add	Depends on where it is in the user interface	<p>Click this button to add an object. It appears in many places in the ElectricFlow UI.</p> <p>Examples:</p> <p>It appears in lower right corner of an application tier. Click it to add a component to the application tier.</p> <p>It also appears in a new resource in an environment tier. Click it to define the resource.</p>








Button or Icon	Name	Tooltip (Hover over the button to view the tooltip.)	How to Use It
	Add +	<p>Depends on where it is in the user interface.</p> <p>In the Applications List: Add new application</p> <p>In the Environments List: Add new environment</p>	Click this button to add an application or environment to the Applications List or the Environments List.
 	Add process	<p>Depends on where it is in the user interface.</p> <p>In the Applications List: Add application process</p> <p>In a component: Add component process</p>	Click this button to add an application or component process.







Button or Icon	Name	Tooltip (Hover over the button to view the tooltip.)	How to Use It
	Add step	Add step	Click this to add a step to an application or component process. When you click it, an undefined step appears. Drag and drop the step to where you want to add it in the process.
	Add tier	Add tier	Click this button to add a tier in the Applications Visual Editor or the Environments Visual Editor.
	Add tier map	Add tier map	Click this button to add a tier map.
	Application	–	When you see this icon in the ElectricFlow UI, the object is an application.
	Applications	–	Click this launch pad to go to the Applications List.
 	Artifact version	–	This appears in the Environment Inventory. The number next to the icon is the artifact version number.





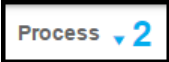
Button or Icon	Name	Tooltip (Hover over the button to view the tooltip.)	How to Use It
	Automations	–	Click this launch pad to go to the automation platform for more details about objects.
	Available	–	<p>When you see this button in the ElectricFlow UI, the environment is available. You can deploy an application in this environment.</p> <p>When you click this button, the environment becomes unavailable, and you cannot use this environment to deploy an application.</p>
	Breadcrumb	–	<p>Breadcrumbs show the path to where you are in the ElectricFlow UI in this format:</p> <p><i>object type/object name/...</i></p> <p>In this example, the breadcrumb goes to a component called <i>hc-warfile</i> and is in this format:</p> <p><i>Applications/<application name>/<component name>.</i></p>
		Change Alert	When you compare snapshots, this icon appears next to objects that have changed.


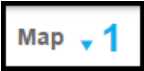





Button or Icon	Name	Tooltip (Hover over the button to view the tooltip.)	How to Use It
	Changes	Changes	<p>The tooltip message varies depending on the object that changed.</p> <p>For example, the tooltip can be "Component property has changed" or "Artifact version has changed."</p>
 	Compare snapshots	Compare snapshots from a list	<p>This is available (enabled) when the Snapshot List has two or more snapshots.</p> <p>Click this to open a full-screen window and view two snapshots next to each other.</p> <p>The default is to have the most recent snapshot on the left and the previous snapshot on the right.</p> <p>Go to Snapshot List on page 405 for more information.</p>
	Component	–	When you see this icon in the ElectricFlow UI, the object is a component.
	Copy		<p>Click this button to copy the information about an object.</p> <p>You can paste it in a text field or area.</p>








Button or Icon	Name	Tooltip (Hover over the button to view the tooltip.)	How to Use It
	Delete	Delete	Click this button to delete the selected objects. The button is enabled when the objects have been selected.
	Edit	Edit	Click this button to edit the name and description of the object.
	Email notifications	Email notifications	Click this to configure email notifications for the selected application.
	Add email notifications	Email notification, add	Click this to add an email notification to the selected application, application process, or process step in the "Application notification / edit" dialog box.
	Delete email notifications	Email notification, delete	Click this to delete an email notification in the selected application, application process, or process step in the "Application notification / edit" dialog box.




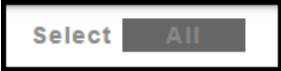


Button or Icon	Name	Tooltip (Hover over the button to view the tooltip.)	How to Use It
	Environment	–	This environment has only static resources and is available.
			This environment has only static resources and is not available.
			This environment has only dynamic resources and is available.
			This environment has only dynamic resources and is not available.
			This environment is a mixed and is available. Some tiers have dynamic resources while others have static resources.
			This environment is a mixed and is not available. Some tiers have dynamic resources while others have static resources.
	Environment s	–	Click this launch pad to go to the Environments List.







Button or Icon	Name	Tooltip (Hover over the button to view the tooltip.)	How to Use It
	Expand	Expand	Click this button to see all the changes to an object in the Change History.
	Export	Export	Click this button to export the object changes and save them as an XML file.
	Help	–	Click this button to see the Help text .
	Hide	Depends on where it is in the user interface. In the Applications List: Hide running process	Click this button to show less details about the object.
	Inventory	Inventory	Click this button to open the environment inventory.
	Lookup	–	Click this button to open the Parameter Lookup dialog box.





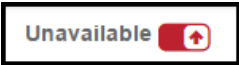
Button or Icon	Name	Tooltip (Hover over the button to view the tooltip.)	How to Use It
	Menu (The tooltip depends on where it is in the user interface.)	In the upper left corner of the page, it is referred to as the <i>Main menu</i> .	When you click this button, a destination list opens that includes the launch pads on the Home page.
		In the upper right corner of a visual editor, it is referred to as the <i>Context menu</i> .	When you click this button, a list of options appears. Click one of these to view more information about the object.
		In the upper right corner of a tier, it is referred to as a <i>Context menu</i> .	When you click this button, a list of options appears. Click one of these to view more information about the Application or Environment tier.
		In the upper left corner of a component or resource, it is referred to as a <i>Context menu</i> .	When you click this button, a list of options appears. Click one of these to view more information about the component or resource.
	Number of processes	–	Click the down arrow next to the number to open the list of existing processes. This also shows the number of application processes for the application.




Button or Icon	Name	Tooltip (Hover over the button to view the tooltip.)	How to Use It
	Number of process steps	–	In the ElectricFlow UI, this icon shows the number of steps in the process.
	Number of tier maps	–	Click the down arrow next to the number to open the list of existing tier maps. This also shows the number of tier maps for the application.
	Number of tiers	–	In the ElectricFlow UI, this icon shows the number of tiers in an application or environment.
	Parameter	–	When you see this icon in the ElectricFlow UI, the object is a parameter.
 	Process	–	When you see this icon in the ElectricFlow UI, the object is an application or component process.
	Process step	–	When you see this icon in the ElectricFlow UI, the object is a step in a component or application process.



Button or Icon	Name	Tooltip (Hover over the button to view the tooltip.)	How to Use It
	Provision	–	When you click this button, you provision cloud resources in a resource templates.
  	Resource	–	<p>When you see this icon in the ElectricFlow UI, the object is a resource.</p> <p>It appears in the Environments List and Environments Visual Editor.</p>
			<p>When you see this icon in the ElectricFlow UI, the object is a resource assigned to an environment tier.</p> <p>It is defined and managed in the automation platform.</p>
	Resource pool		When you see this icon in the ElectricFlow UI, the object is a resource pool.
	Revert	Revert	Click this button to revert the object to a previous version.

Button or Icon	Name	Tooltip (Hover over the button to view the tooltip.)	How to Use It
	Run process	Run process	Click this button to deploy (run) the application. The button is enabled when it is green.
	Schedule	Schedule	Click this button to add or view schedules for an application.
	Search	Launch Change History Search	Click this button to open the Change History - Search dialog box.
 	Select	–	Click this button to toggle between All and None . Click All to go to None . Click None to go to All . When All appears, all the objects in the list are selected. When None appears, no objects are selected.
	Select All	Select All	Click this button to see all the changes to an object in the Change History.

Button or Icon	Name	Tooltip (Hover over the button to view the tooltip.)	How to Use It
	Snapshot	Deploy snapshot	<p>Click this button to create a snapshot (New Snapshot) or go to the Snapshot List (Snapshot List).</p> <p>When you see this icon in the Snapshot dialog boxes, the object is a snapshot.</p>
	Tear down	–	<p>Click the button to retire a dynamic environment. This button works only on dynamic environments.</p>
	Template	–	<p>Depending on where you click the button, clicking this button results on one of these actions:</p> <ul style="list-style-type: none"> • Display the history of the environment template in the Environments List. • Open a drop-down list of templates that you can use for an email notification setting. • Open the Environment Templates List. • Open the Resource Templates List.
	Email message template		
	Environment template		
	Resource template		

Button or Icon	Name	Tooltip (Hover over the button to view the tooltip.)	How to Use It
	Tier	–	When you see this icon in the ElectricFlow UI, the object is an environment tier.
	Tier map	–	When you see this icon in the ElectricFlow UI, the object is a tier map.
 	Track Changes	<p>Depends on where it is in the user interface.</p> <ul style="list-style-type: none"> • Launch Change History Search • Track Changes 	<p>Click this button to open the Change History for an object.</p> <p>When you hover over this icon, this tooltip appears: Track Changes.</p>
	Unavailable	–	<p>When you see this button in the ElectricFlow UI, the environment is not available. You cannot deploy an application in this environment.</p> <p>When you click this button, the environment becomes available, and you can deploy an application in it.</p>

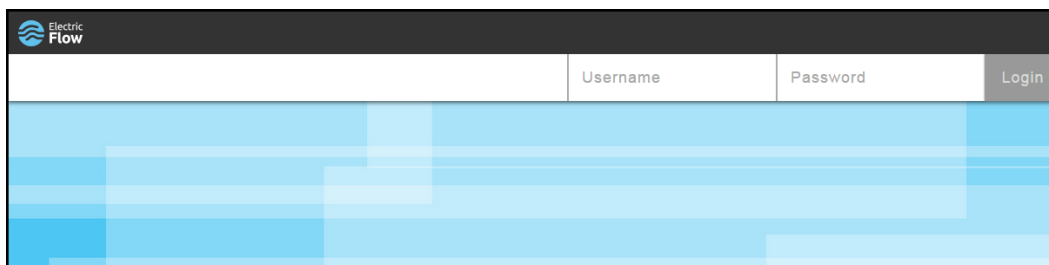
Button or Icon	Name	Tooltip (Hover over the button to view the tooltip.)	How to Use It
	View	Depends on where it is in the user interface. In the Applications List: View running process	Click this button to show more details about the object.
	View details	View details	Click this button to view more details about the object. Depending on where you are in the ElectricFlow UI, you may go to a new page or dialog box.
	View details	–	Click this button to open the automation platform. A page opens where you can view more details about the object in the automation platform.

Button or Icon	Name	Tooltip (Hover over the button to view the tooltip.)	How to Use It
	View only information	–	<p>This information is view only.</p> <p>ElectricFlow automatically adjusts the page settings to show all the information on the page.</p> <p>For example, if application tiers do not fit on the page at 100% magnification, ElectricFlow reduces the magnification until they all appear on the page.</p>
	View path	View path	Click this button to see the path to the object in the change history.

Landing Page

How to get to here: Enter **http://<ElectricFlow-server>/flow** in a browser window, where *<ElectricFlow-server>* is the ElectricFlow server IP address or host name.

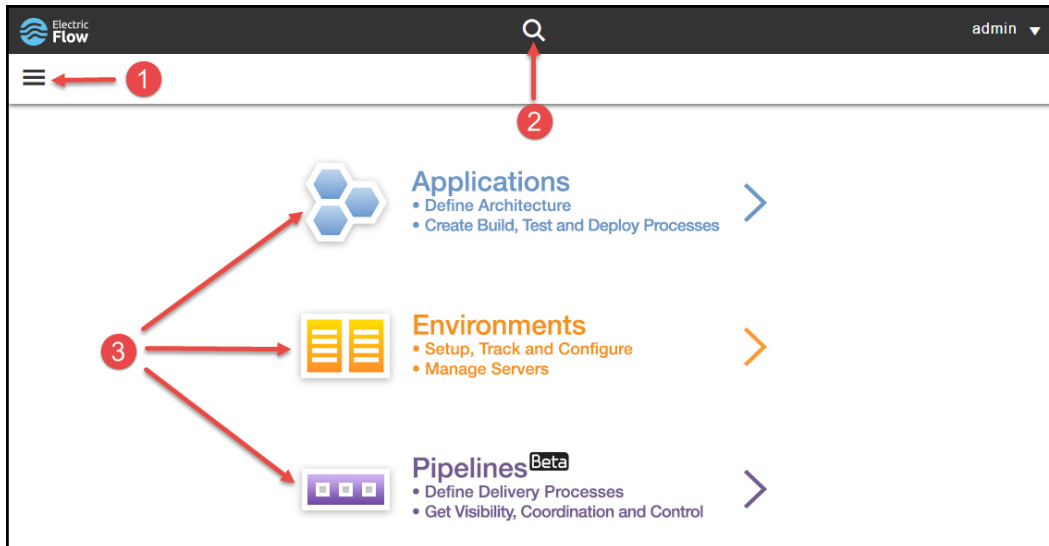
To log in, go to the landing page:



Home Page

How to get to here: From the landing page, enter your user name and password and click **Sign in**.

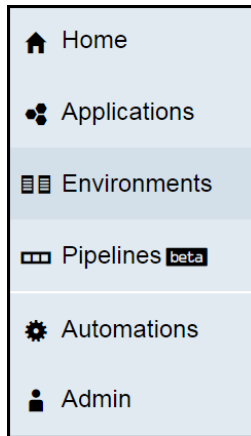
From the Home page, you can model an application for your continuous delivery solution.



1	Menu button —Click this to open a list of destinations, which includes the list of launch pads on this page.
2	Search button —Click this to open the Change History - Search dialog box.
3	<p>Launch pads—Click one of these buttons to start modeling the ElectricFlow application.</p> <ul style="list-style-type: none"> • Applications—Click this to open the Applications List. From this list, you start to model applications by defining the application architecture and modeling the component and application processes. • Environments—Click this to open the Environments List. From this list, you start to model the environments in which applications are run by assigning and managing resources. • Pipelines—Click this to open the Applications List. From this list, you start to model pipelines.

Main Menu

How to get here: Click on the **Main menu** button.

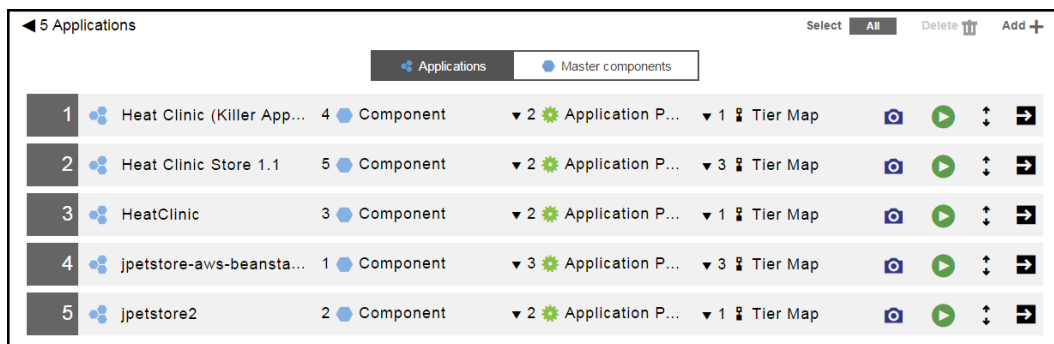


These are the main menu options:

- **Home**—Click **Home** to close this menu and return to the Home page.
- **Applications**—Click **Applications** to open the Applications List.
- **Environments**—Click **Environments** to open the Environments List, Environment Templates List, or the Resource Templates List.
- **Pipelines**—Click **Pipelines** to open the Pipelines List or Pipeline Runs List.
- **Automations**—Click **Automations** to open the Procedures, Workflows, Artifacts, Credentials, or Platform/Projects page.
- **Admin**—Click **Admin** to open the Users, or Groups page in the ElectricFlow automation platform.

Applications

When you click **Applications**, the Applications List opens:



Environments

When you click **Environments**, a menu where you select the type of environment to view or create opens.

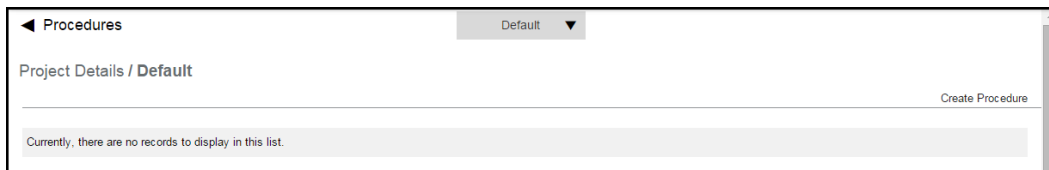
- When you click **Environments > All Environments**, the Environments List opens.
- When you click **Environments > Environment Templates**, the Environment Templates List opens.

- When you click **Environments > Resource Templates**, the Resource Templates List opens.
- When you click **Environment > Resources**, the Resource page opens.

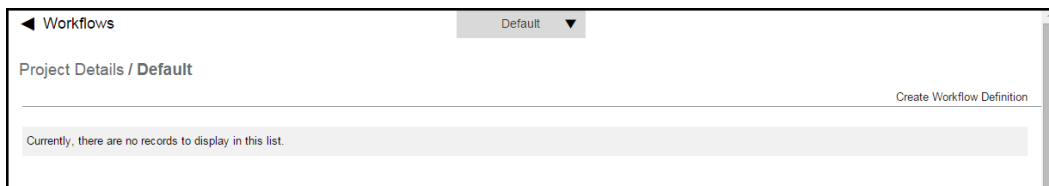
Automations

When you click **Automations**, a menu opens with following options:

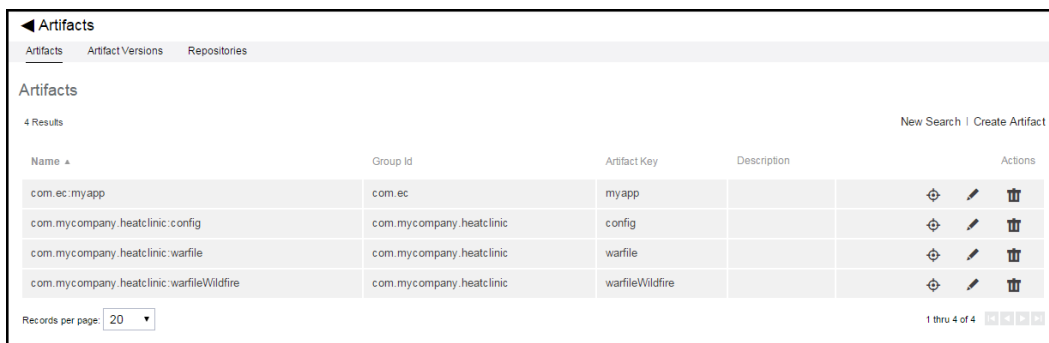
Procedures—The **Procedures** page opens.



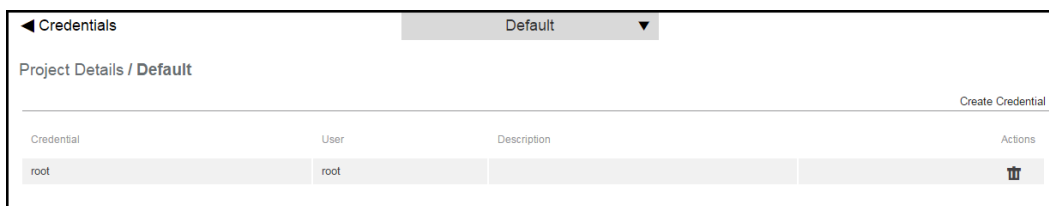
Workflows—The **Workflows** page opens.



Artifacts—The **Artifacts** page opens.



Credentials—The **Credentials** page opens.



Platforms/Projects—The **Projects** page opens.

Home

Flow Tools

Projects

Jobs

Workflows

Cloud

Artifacts

Search

Administration

Change History

10 Results

All Projects

Create Project

New Search

Project	Description	Impersonation Credential	Create Date	Actions
Default	Default project created during installation.		2015-07-10 12:56:39 PDT	<div></div> <div></div> <div></div> <div></div>
DemoLibrary	Library of procedures used by the ElectricFlow Demo System.		2015-06-12 00:56:23 PDT	<div></div> <div></div> <div></div> <div></div>
DeployObjects-1.7.8			2015-07-10 13:02:26 PDT	<div></div> <div></div> <div></div> <div></div>
EC-Examples	This project contains templates for procedures that perform basic tasks within ElectricCommander.		2015-07-10 12:56:40 PDT	<div></div> <div></div> <div></div> <div></div>
EC-Tutorials-1.0.0.69904	This project contains small pieces of code which show how to use features in ElectricCommander.		2015-07-10 12:57:51 PDT	<div></div> <div></div> <div></div> <div></div>
EC-Utilities	This project contains procedures that perform basic tasks within ElectricCommander. Also, the procedures can be used as examples that can be copied and modified in another project.		2015-07-10 12:56:50 PDT	<div></div> <div></div> <div></div> <div></div>
Electric Cloud	Electric Cloud Procedures		2015-07-10 12:57:18 PDT	<div></div> <div></div> <div></div> <div></div>
beanstalk	AWS procedures to manipulate Amazon resources for <i>Elastic Beanstalk</i> operations: <ul style="list-style-type: none"> S3 storage for file deployments Elastic Beanstalk Environments Elastic Beanstalk Applications 		2015-01-06 06:47:18 PST	<div></div> <div></div> <div></div> <div></div>
ec-library	Library of Electric Cloud utility procedures.		2015-01-07 05:55:23 PST	<div></div> <div></div> <div></div> <div></div>
jpetstore			2015-02-19 05:06:40 PST	<div></div> <div></div> <div></div> <div></div>

Records per page:

20

1 thru 10 of 10

Admin

When you click **Admin**, a menu opens with following options:

Users—The **Users** page opens. You can view and manage the ElectricFlow users from this page.

Users

Create Local User

Filter:

Maximum Results:

Include Inactive Users: ☐

OK

Groups—The **Groups** page opens. You can view and manage the ElectricFlow groups from this page.

Groups

Create Local Group

Filter:

Maximum Results:

Include Inactive Groups: ☐

OK

Plugins—The **Plugins** page opens. You can view and manage the ElectricFlow plugins from this page.

◀ Plugins

Plugin Manager

Currently Installed | Install from File/URL | View Catalog

Category: All Show Promoted Plugins Only: ☐



Plugin Label	Author	Version	Category	Description	Actions
DeployObjectViewer	Electric Cloud	2.1 (promoted)	System	This plugin shows Electric Commander Deploy objects	Demote Help Uninstall
DeployObjects	Greg Maxey	1.7.8 (promoted)	System	This projects allows creation of Deploy objects through a JSON model.	Demote Help Uninstall
EC-ALM	Electric Cloud	1.0.7.93709 (promoted)	Test	Integrates HP-ALM Test server into Electric Commander	Demote Configure Help Uninstall
EC-AgentManagement	Electric Cloud	1.1.0.94286 (promoted)	System	Centralized Agent Management procedures	Demote Help Uninstall
EC-Ant	Electric Cloud	2.0.9.93709 (promoted)	Build	Java build tool	Demote Help Uninstall
EC Artifact	Electric Cloud	1.1.1.93709 (promoted)	System	Provides an interface for artifact version publish and retrieve operations	Demote Uninstall

Licenses—The **Licenses** page opens. You can view and manage the ElectricFlow plugins from this page.

◀ Licenses

Licenses

All Licenses Import License

Company	Feature	Expiration	Grace Period (Days)	Actions
Electric Cloud	Server (enterprise)	2016-01-07	0	 

Current Usage

Maximum Concurrent Steps: unlimited

Concurrent Steps: 0

Maximum Concurrent Hosts: 100

Concurrent Hosts In Use: 0

Maximum Provided Hosts: 100

Provided Hosts In Use: 0

Maximum Concurrent Users: unlimited

Active Users: 0

Server—The **Server** page opens. You can view and manage the access control settings from this page.



◀ Server

Server Settings Access Control

System Access Control

Category	Description
Administration	Read permission allows access to the getStatus, getDatabaseConfiguration[s], and export(global) API functions. Modify permission allows access to the shutdown, setDatabaseConfiguration, and import(global) API functions.
Artifacts	Read permission allows access to the getArtifact[s] API functions. Modify permission allows access to createArtifact and deleteArtifact API functions.
Directory	Read permission allows access to the getUsers, getGroups, and getDirectoryProviders API functions. Modify permission allows access to the createUser, createGroup, deleteUser, deleteGroup, createDirectoryProvider, modifyDirectoryProvider, deleteDirectoryProvider, testDirectoryProvider, and moveDirectoryProvider API functions.
Email Configurations	Modify permission allows access to the createEmailConfig and deleteEmailConfig API functions.
Force Abort	Execute permission allows access to the force option on the abortJob API function.
Licensing	Read permission allows access to the getLicense[s] API functions. Modify permission allows access to the importLicenseData and deleteLicense API functions. Execute permission allows access to the preemptLicense API function.
Logging	Read permission allows access to all entries in the event log regardless of their container. Modify permission allows deletion of event log entries and access to the logMessage API function.
Plugins	Read permission allows plugins to be used. Modify permission allows access to the installPlugin, promotePlugin and uninstallPlugin API functions.
Priority	Execute permission allows the user who launches a procedure using the runProcedure API function to raise the priority of the job.
Projects	Modify permission allows access to the createProject and deleteProject API functions.
Repositories	Read permission allows access to the getRepository API function. Modify permission allows access to the createRepository, deleteRepository, modifyRepository and moveRepository API functions.
Resources	Modify permission allows access to the createResource and deleteResource API functions.
Session	Execute permission allows access to the login API function.
Workspaces	Modify permission allows access to the createWorkspace and deleteWorkspace API functions.
ZonesAndGateways	Modify permission allows access to the createZone, deleteZone API functions. Modify permission also allows access to deleteResource API function when the resource belongs to a Gateway.

Custom Server Properties Create Property | Create Nested Sheet | Access Control

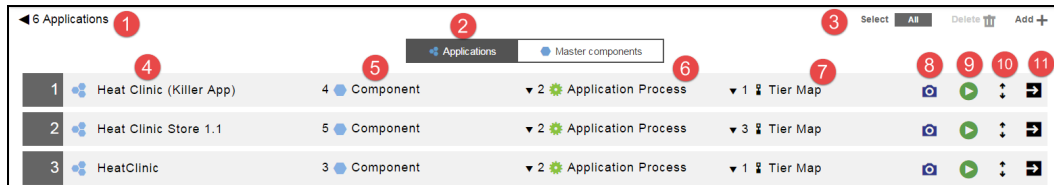
Property Name	Value	Description	Actions
Electric Cloud			 

Applications List

How to get here:

One of these ways:

- From the Home page, click the **Applications** launch pad.
- Click the **Main menu** button, and then click **Applications**.

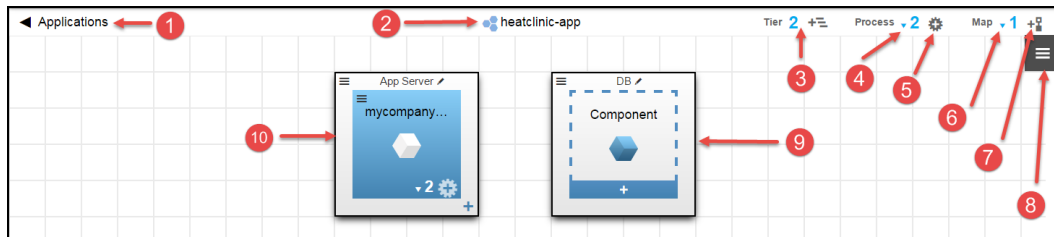


1	Breadcrumb showing the total number of applications in your ElectricFlow system.
2	You can view a list of the applications or the master components in your system. Select the Applications tab to view the list of applications.
3	Click the Select button to select all or none of the items in the list. Click the Delete button to delete the selected items. Click the Add + button to add an application.
4	Click the application name (in this example, Heat Clinic (Killer App)) to go to the Applications Visual Editor.
5	Number of components in the application.
6	Number of application processes in the application.
7	Number of tier maps for the application.
8	Click the Snapshot button to take a new snapshot when you deploy (run) the application or want to add a snapshot to the Snapshot List.
9	Click the Run process button to deploy (run) the application when the button is green. This is enabled when the application has one or more application processes.
10	Click the View button to see more details of the running process.
11	Click the View details button to go to the Applications Visual Editor.

Applications Visual Editor

How to get here: From the Applications List, click an application name.

Applications consist of application processes and components grouped into tiers.



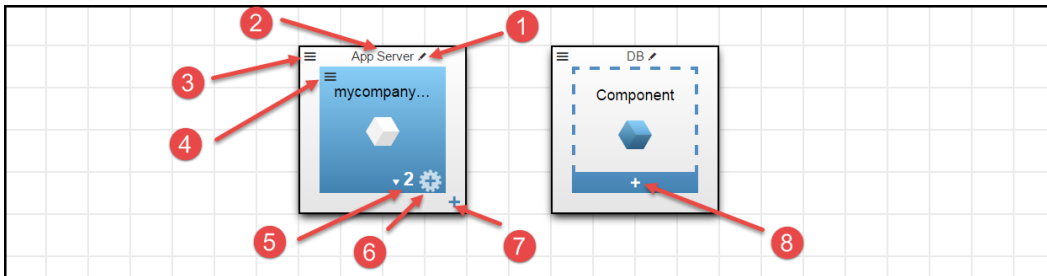
1	Breadcrumb to the application
2	Name of the application
3	Number of tiers and the Add tier button Click the Add tier button to add a tier.
4	Click the number of applications (2) button to view the list of application processes for the application.
5	Click the Add process button to add a process.
6	Click the number of tier maps (1) button to view the list of tier maps for the application.
7	Click the Add tier button to add a tier map.
8	Click the Menu button to open the context menu. You can get more details about the application: <ul style="list-style-type: none"> • Details—The name and description of the object. • Properties—The properties in the object. • Notifications—The email notifications configured for the object. • Access Control—The access control configuration in the automation platform for the object. • Track Changes—The change history of the object. • Delete—Delete this object.
9	Application tier with an undefined component
10	Application tier with a defined component

Application Tiers

How to get here: From the Applications Visual Editor, choose an application tier.

These are application tiers.

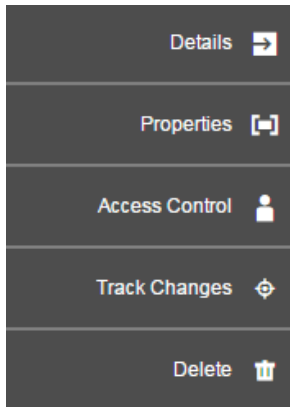
- The "App Server" tier has a defined component with two component processes.
- The DB tier has an undefined component.



1	Click the Edit button to edit the tier name and description in the Application Tier Details dialog box.
2	Application tier name
3	Click the Menu button to view the application tier details.
4	Click the Menu button to view the component details.
5	Click the 2 (number of component processes) button to view the component processes.
6	Click the Add process button to add a component process.
7	Click the + button to add a component to the application tier.
8	In an undefined component, click the + button to define the component.

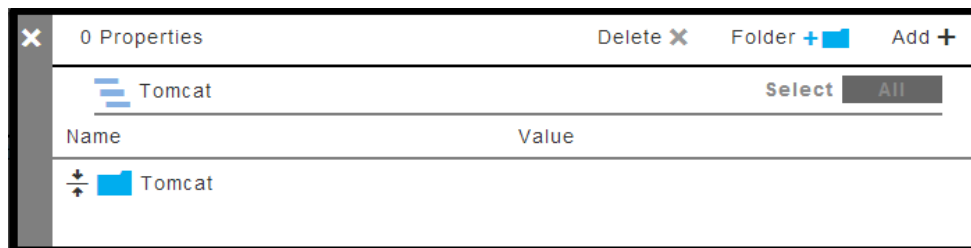
Context Menu for the Application Tier and Component

This menu appears when you click the **Menu** button:



In the Application Tier

- When you click **Details**, the **Application Tier Details** dialog box appears.
You can edit the application name and description.
- When you click **Properties**, the **Properties** dialog box appears, where you can set the properties for the application tier.



- When you click **Access Control**, you go to the Access Control page for the tier in the automation platform, where you can set privileges for the objects in your application.
- When you click **Track Changes**, the Change History Search Form opens.
- When you click **Delete**, the **Delete Application Tier** dialog box appears.
Click **OK** to verify that you want to delete this application tier.

In the Component

- When you click **Details**, the **Component Details** dialog box appears. The information that appears depends on your system.

In this example, the **Content Location** is EC-Artifact, the default plugin. In the **Artifact** field,

com.mycompany.heatclinic:config comes from the Artifact Repository in the automation platform.

Component Details

database.conf Description

EC-Artifact Browse

Artifact: Required

Version: ☒ Latest

☐ Exact:

☐ Range:

Minimum: ☐ Inclusive?

Maximum: ☐ Inclusive?

Retrieve to directory: ☒ Overwrite:

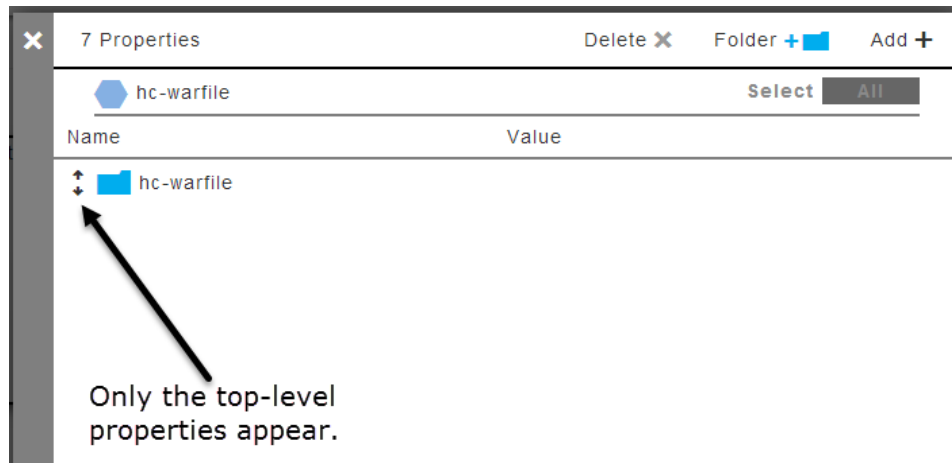
Retrieved Artifact Location Property:

Cancel OK

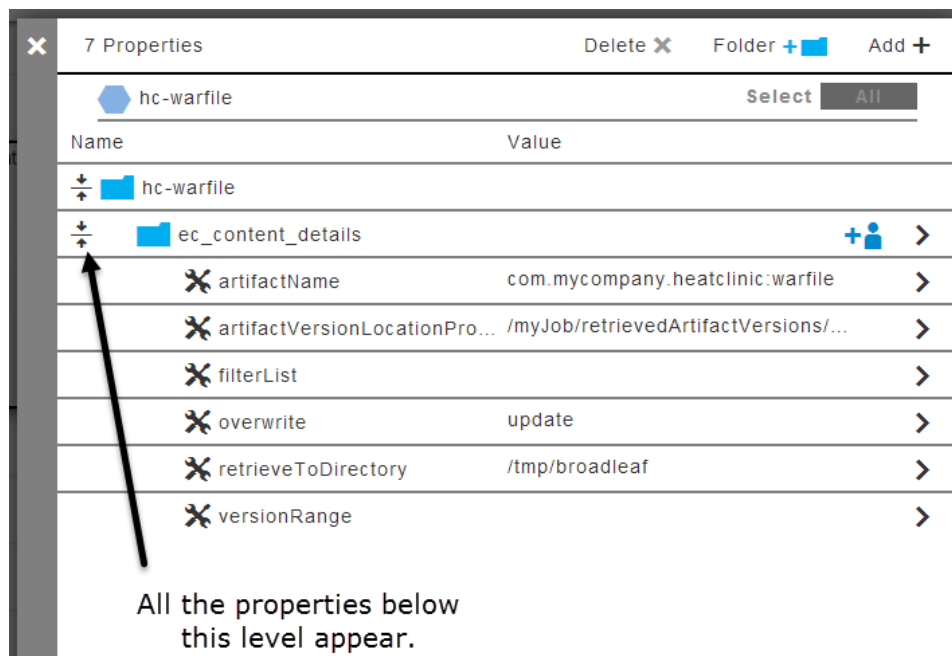
1	Content Location of the component
2	Component details that vary depending on the Content Location

- When you click **Properties**, the component **Properties** dialog box appears. You can set the properties for the component.

In this example, only the top-level component details appear.



When you click the **View** button, all the properties below the current level appear.



You can click the **Hide** button to show less details.

- When you click **Access Control**, the Access Control page for the component in the automation platform opens, where you can set privileges for the objects in your application.

For more information, go to the automation platform Help > **Overview** > **Access Control**.

- When you click **Track Changes**, the Change History Search Form opens.
- When you click **Delete**, the **Component Deletion** dialog box opens.

Click **OK** to verify that you want to delete this component.

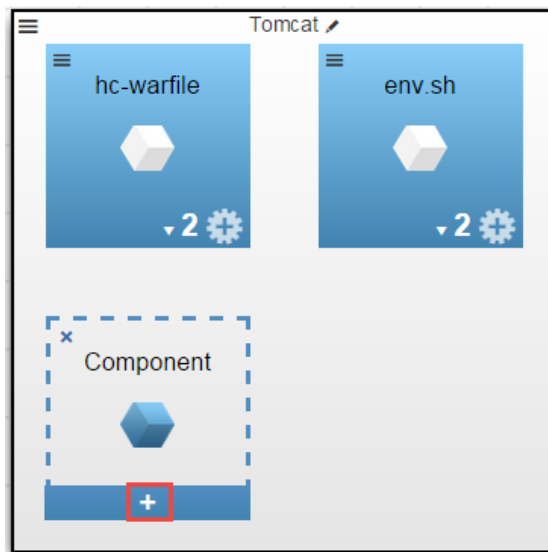
Defining Components

This example shows how to define a component in an application tier.

How to get to here: From the Applications Visual Editor, choose an application tier.

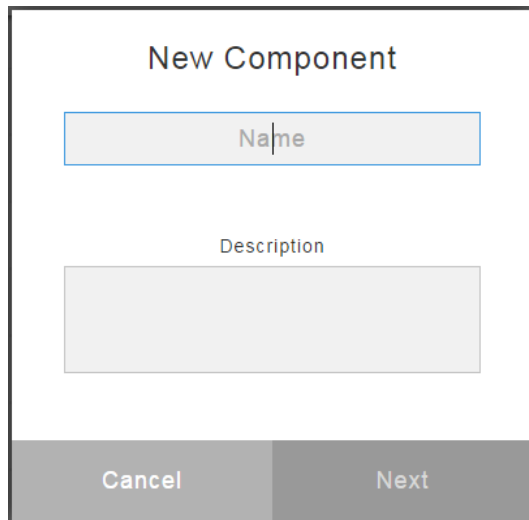
1. Click the **Add component** button to add a component.

A new component appears in the application tier.



2. Click the **+** button to define the component.

The **New Component** dialog box appears.

A dialog box titled "New Component" with a light gray background and a dark gray border. It contains two input fields: a text field labeled "Name" with a blue border and a cursor, and a larger text area labeled "Description" with a light gray border. At the bottom, there are two buttons: "Cancel" and "Next", both with a dark gray background and white text.

New Component

Name

Description

Cancel Next

3. Enter the name and optional description of the component and click **Next**.

The **Component Details** dialog box appears.

4. Select the **Content Location**, enter the component details, and click **OK**.

In the following example:

- The new component is named *database.conf*.
- The Content Location is *EC-Artifact*, the default plugin.
- In the **Artifact** field, *com.mycompany.heatclinic:config* comes from the Artifact Repository in the automation platform.

Component Details

database.conf

Description

1

EC-Artifact

Browse

Artifact: com.mycompany.heatclinic:config Required

Version: ☒ Latest

☐ Exact:

☐ Range:

Minimum: ☐ Inclusive?

Maximum: ☐ Inclusive?

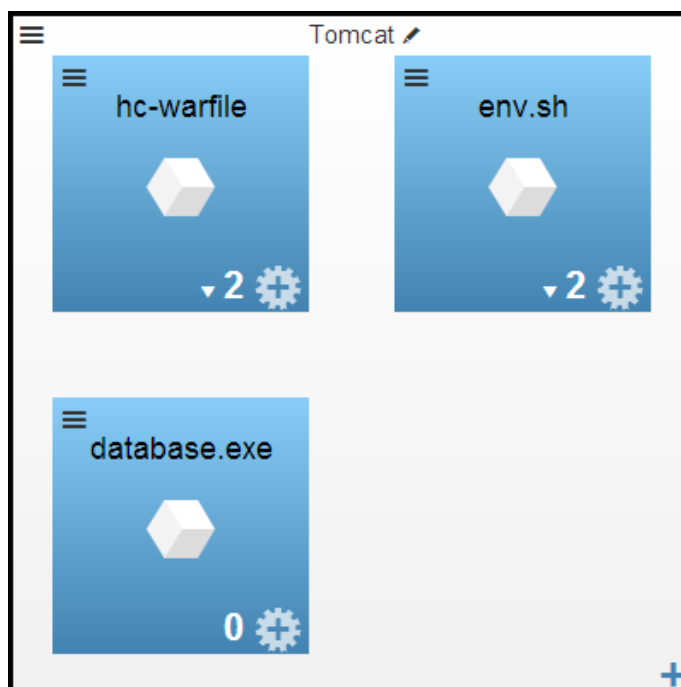
Retrieve to directory: ☒ /tmp/broadleaf Overwrite: update

Retrieved Artifact Location Property: /myJob/retrievedArtifactVersions/\${assignedResourceName}

Cancel OK

1	Content Location of the component
2	Component details that vary depending on the Content Location

The Applications tier now has new defined component.



Creating Component and Component Processes

Conceptual information about component processes and the top 10 guidelines when using them.

You can configure component process steps with one of the following:

Artifacts

Component processes

Defining process steps: Plugins, commands, etc.

Application processes

Credentials and impersonation

Drop and drag to add a new step

Process branching

These methods are the same as those that you can use for application process steps.

Component Processes

How to get to the Component Process Visual Editor when modeling a new component process:

From the Applications Visual Editor, select a component in an application tier, and click the **Add process** button.

The **New Component Process Details** dialog box appears.

New

Component Process Details

Name

Description

Deploy
▼

Credential
0
Optional >

Workspace

default
▼

Optional

Time limit

0

Seconds

▼

Optional

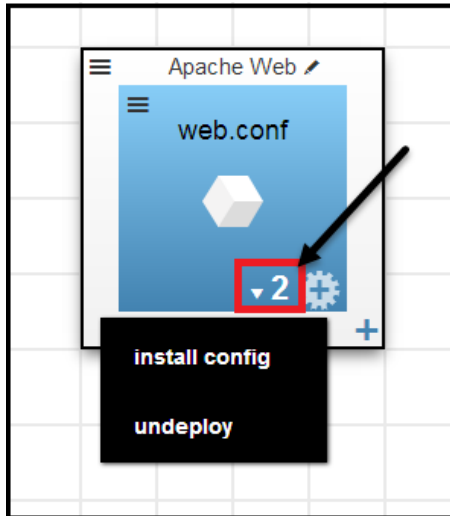
Cancel
OK

Field	Description and How to Set It
Name	Name of the process step
Description	Description of the process step
Process Type	<p>Type of process. The default is Deploy.</p> <p>To set the process type:</p> <ol style="list-style-type: none"> 1. Click the Type field to select the process type. 2. Select one of these options: <p>Deploy—Enables Inventory Tracking. The ElectricFlow server tracks the artifacts deployed to environments.</p> <p>Undeploy—The next time that the process is run, the ElectricFlowserver removes information about the artifacts deployed to environments.</p> <p>Other—Disables Inventory Tracking.</p>

Field	Description and How to Set It
Credential	<p>An object consisting of a user name and password that ElectricFlow uses to run a process step.</p> <p>The dialog box displays the number of credentials for the process step, which are the same credentials that you use with procedures, steps, and schedules in the automation platform.</p> <p>You can only impersonate one credential. See Adding Credentials on page 232 to set the process type.</p>
Workspace	<p>Area in the disk space where the files and results of the job step are stored.</p> <p>To set the workspace, click the Workspace field to open a drop-down list of workspaces in the ElectricFlow platform and select a workspace.</p> <p>For more information about workspaces, go to the "Workspaces and Disk Management" topic. To set the workspace, click Workspace to open a drop-down list of workspaces in the ElectricFlow platform. Select a workspace, and click OK.</p>
Time limit	<p>Maximum length of time that the step is allowed to run. After the time specified, the step is aborted,</p> <p>To set the time limit, enter the time and select the unit of time: seconds, minutes, or hours.</p> <p>For information about time limits for procedure job steps in the ElectricFlow platform, go to the "Job Step Details" topic.</p>

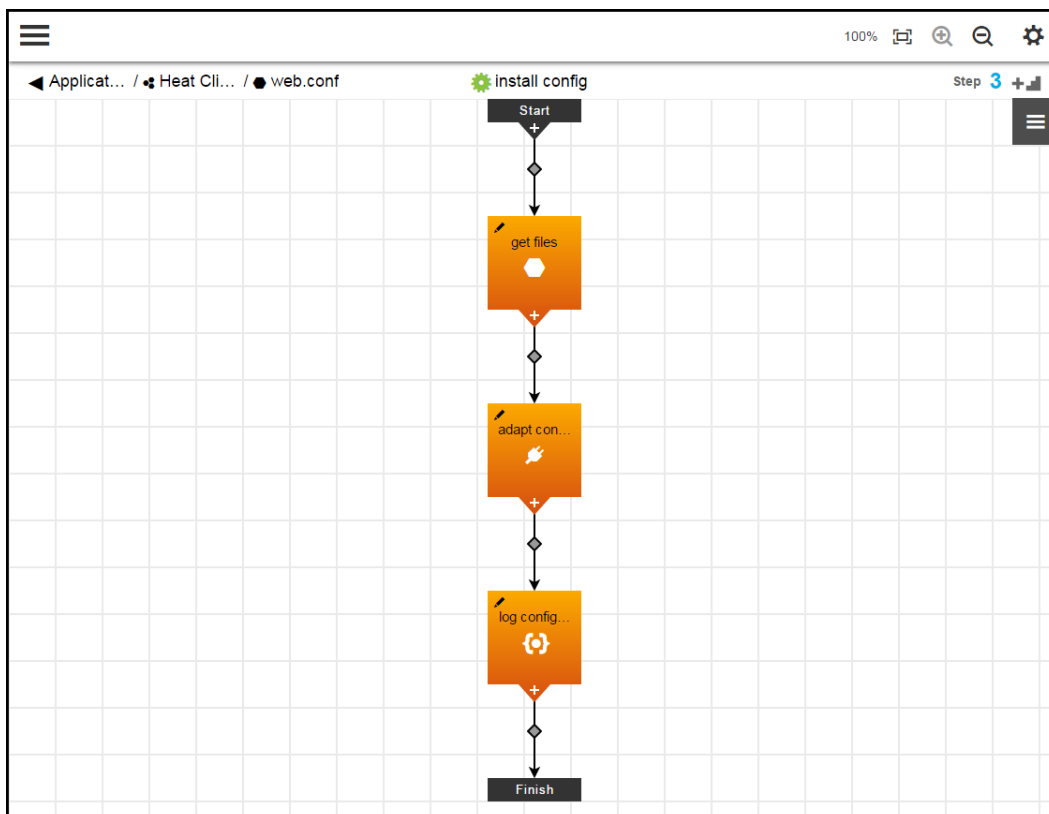
How to get to the Component Process Visual Editor after selecting an existing component process:

From the Applications Visual Editor, select a component in an application tier, click the **2** (number of component processes) button, and then select a component process in the drop-down list.

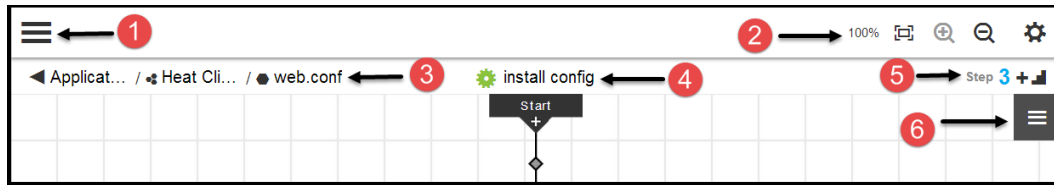


Component Process Visual Editor UI

The steps in the component process called "install config" appear in the Component Process Visual Editor.

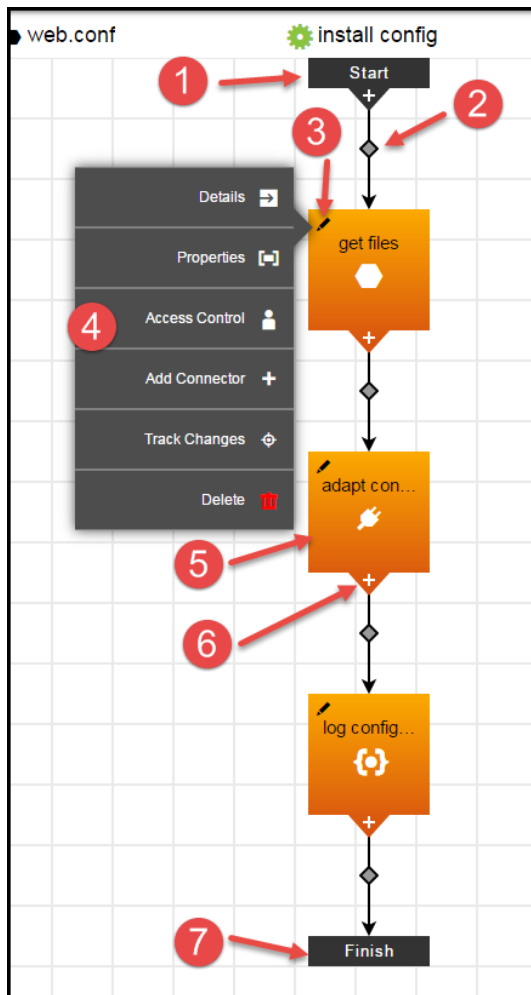


The Component Process Visual Editor has these objects:



1	Main menu
2	<p>This information is view only.</p> <p>ElectricFlow automatically adjusts the page settings to show all the tiers in the application.</p> <p>For example, if all the tiers do not fit on the page at 100% magnification, ElectricFlow reduces the magnification until all the tiers appear on the page.</p>
3	Breadcrumbs specifying the path to the component: <i>object type/application name/component name</i>
4	Name of the component process
5	Number of steps in the process and the Add step button
6	<p>Click the Menu button to view the component process details.</p> <ul style="list-style-type: none"> • Details—The name and description of the object. • Properties—The properties in the object. • Access Control—The access control configuration in the automation platform for the object. • Track Changes—The change history of the object. • Delete—Delete this object.

The component process has these objects:



1	Start of the process
2	Click this Connector button to configure the branching conditions between two process steps. The default is Always , which means always go to the next step.
3	Click the Edit button (a pencil) to open the process-step details menu. <ul style="list-style-type: none"> • Details—The name and description of the object. • Properties—The properties in the object. • Access Control—The access control configuration in the automation platform for the object. • Add Connector—Add a connector from the selected step. • Track Changes—The change history of the object. • Delete—Delete this object.

4	Process-step details menu
5	Component process step
6	Click the + button to add a step below the current step.
7	End of the process

Application Processes

How to get to the Application Process Visual Editor when modeling a new application process:

From the Applications Visual Editor, click the **Add process** button.

The **New Application Process Details** dialog box appears.

Field	Description and How to Set It
Name	Name of the process step
Description	Description of the process step

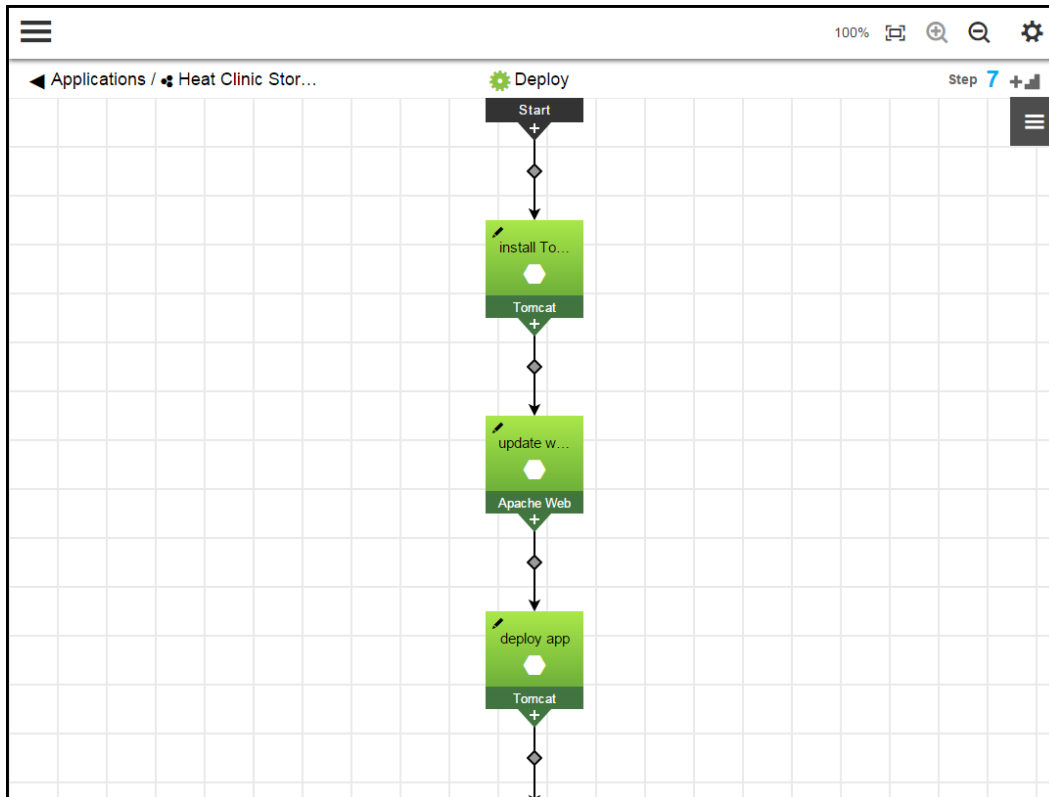
Field	Description and How to Set It
Credential	<p>An object consisting of a user name and password that ElectricFlow uses to run a process step.</p> <p>The dialog box displays the number of credentials for the process step, which are the same credentials that you use with procedures, steps, and schedules in the automation platform.</p> <p>You can only impersonate one credential.</p>
Workspace	<p>Area in the disk space where the files and results of the job step are stored.</p> <p>To set the workspace, click the Workspace field to open a drop-down list of workspaces in the ElectricFlow platform and select a workspace.</p> <p>For more information about workspaces, go to the "Workspaces and Disk Management" topic.</p> <p>To set the workspace, click Workspace to open a drop-down list of workspaces in ElectricFlow, select a workspace, and click OK.</p>
Time limit	<p>Maximum length of time that the step is allowed to run. After the time specified, the step is aborted,</p> <p>To set the time limit, enter the time and select the unit of time: seconds, minutes, or hours.</p> <p>For information about time limits for procedure job steps in ElectricFlow, go to the "Job Step Details" topic.</p>

How to get to the Application Process Visual Editor when selecting an existing application process:

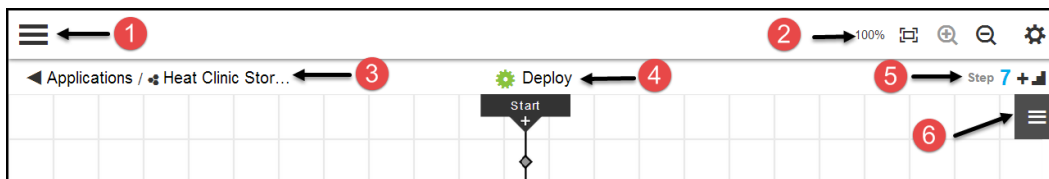
In the Applications Visual Editor, click the down arrow next to the "number of component processes" button, and then select an application process in the drop-down list.

Application Process Visual Editor UI

The steps in the application process called Deploy appear in the Application Process Visual Editor.



The Applications Process Visual Editor has these objects:



1	Main menu
2	<p>This information is view only.</p> <p>ElectricFlow automatically adjusts the page settings to show all the tiers in the application.</p> <p>For example, if all the tiers do not fit on the page at 100% magnification, ElectricFlow reduces the magnification until all the tiers appear on the page.</p>
3	Breadcrumb specifying the path to the application process: <i>object type/application name</i>
4	Name of the application process
5	Number of steps in the process with the Add step button.

6	<p>Click the Menu button to view the application process details.</p> <ul style="list-style-type: none">• Details—The name and description of the object.• Parameters—Opens the Parameter dialog box where you can view, add, and delete the custom parameters for this application process.• Properties—The properties in the object.• Access Control—The access control configuration in the automation platform for the object.• Track Changes—The change history of the object.• Delete—Deletes this object.
---	---

Example: Defining Process Steps

This example describes how to model the process steps in a process visual editor with commands. The procedure to model a component process step in the Component Process Visual Editor is the same as the procedure to model an application process in the Application Process Visual Editor.

- To model steps in an application process, go to the Application Process Visual Editor.

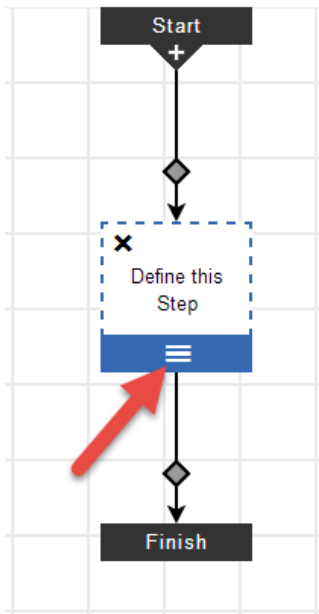
How to get to there: From the Applications Visual Editor, click the **Add process** button, enter the information about the process in the **New Application Process Details** dialog box, and click **OK**.

- To model steps in a component process, go to the Component Process Visual Editor.

How to get to there: From the Applications Visual Editor, select a component, click the **Add process** button, enter information about the process in the **New Component Process Details** dialog box, and click **OK**.

Starting in the appropriate visual editor:

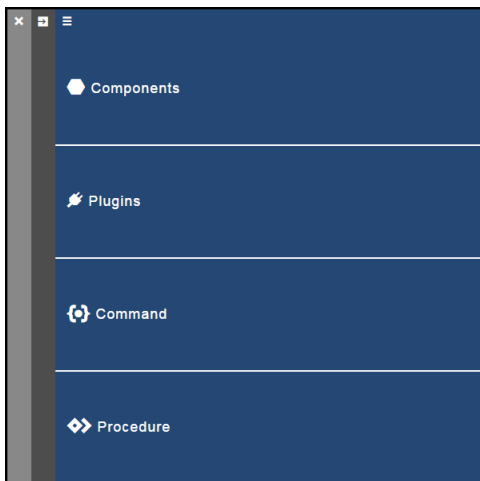
1. Click the button below "Define this Step" to define a process step in the new process.



A dialog box appears.

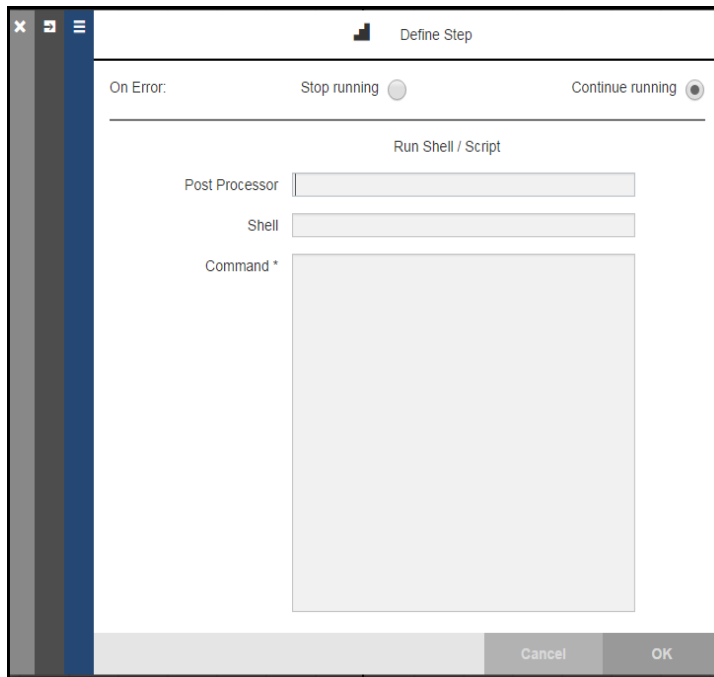
2. Enter information about the process step.
3. Click **Next**.

The process step dialog box appears.



4. Click **Command** to define the process step with a command.

The **Define Step** dialog box appears.



5. Enter the following information:

- Select **Stop running** or **Continue running** in the **On Error** field.
 - When you select **Stop running**, ElectricFlow stops the job if an error occurs.

This step overrides the process branching condition. When an error occurs, the process aborts regardless of the branching condition.
 - When you select **Continue running**, ElectricFlow continues to run the job if an error occurs.

This setting overrides the process branching condition. When an error occurs and the branching condition is Failure, the process continues to the next step.
- Enter the postprocessor command in the **Post Processor** field.
- Enter the shell name in the **Shell** field.
- Enter the command in the **Command** field.

6. Click **OK**.

The step now appears in the process.

7. Click the **+** button below the first step to add another step to the process.
8. Repeat this procedure to define the new step.

Component and Application Process Steps

This topic describes the **Component Process Step** and **Application Process Step** dialog boxes that you use to model component and application process steps.

Designing a New Process Step

How to get to the Component Process Step dialog box: From the Applications Visual Editor, choose an application tier, select a component, and click the **Add process** button.

The screenshot shows a 'New Step' dialog box with the following fields and controls:

- Step name:** A text input field containing 'Name' with a 'Required' label to its right.
- Description:** A large text area for entering a description.
- Credential:** A dropdown menu showing '0' with a right arrow icon.
- Workspace:** A dropdown menu showing 'Select'.
- Time limit:** A text input field containing '0' and a dropdown menu showing 'Seconds'.
- Buttons:** 'Cancel' and 'Next' buttons at the bottom right.

How to get to the Application Process Step dialog box: From the Applications Visual Editor, click the **Add process** button in the upper right corner.

New

Application Process Details

Name

Description

Credential 0 Optional >

Workspace default Optional

Time limit 0 Seconds Optional

Cancel OK

Modeling an Existing Process Step

How to get to the Component Process Step dialog box: From the Applications Visual Editor, choose application tier, select a component, and click the number button (number of component processes).

How to get to the Application Process Step dialog box: On the Applications Visual Editor, click the number button (number of application processes) in the upper right corner.

Setting Parameters in the Process Step Dialog Boxes

IMPORTANT:

When you impersonate a credential, make sure that the impersonated user has the absolute path to the bin directories in the \$PATH environment.

If you define a process step with a command, you must enter the absolute path in the **Post Processor** and **Shell** fields in the Define Step dialog box.

Enter information in the following fields:

- **Name** (required)—Name of the process step.
- **Description**—Description of the process step.
- **Tier** (application process step only)—Application tier in which the process step runs.

- **Credential**—An object consisting of a user name and password that ElectricFlow uses to determine who or what runs a process step.

The field displays the number of credentials for the process step, which are the same credentials that you use with procedures, steps, and schedules in the automation platform.

To set the credentials, click > to open the Credentials dialog box.

You can add only one credential for impersonation, and you can attach more than one credential to the process step.

If a credential is set to impersonate, you can also attach that credential to the process step.

For more information about credentials, go to the automation platform > **Credentials and User Impersonation**.

- **Workspace**—Area in the disk space where the files and results of the job step are stored.

To set the workspace, click the **Workspace** field to open a drop-down list of workspaces in the automation platform and select a workspace.

For more information about workspaces, go to the "Workspaces and Disk Management" topic.

To set the workspace, click **Workspace** to open a drop-down list of workspaces in the ElectricFlow platform, select a workspace, and click **OK**.

- **Time limit**—Maximum length of time that the step is allowed to run. After the time specified, the step is aborted.

To set the time limit, enter the time and select the unit of time: seconds, minutes, or hours.

For information about time limits for procedure job steps in the ElectricFlow platform, go to the "Job Step Details" topic.

Parameters in Application Processes

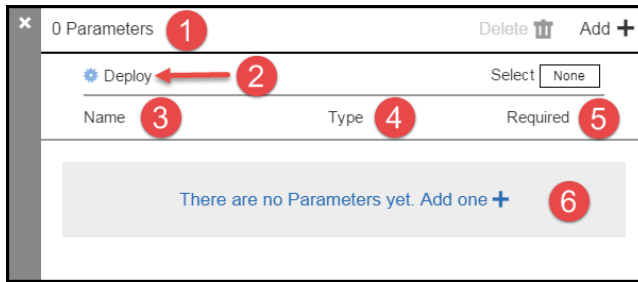
You can define and apply custom parameters to application processes to automate your deployments. You define the parameters and apply them while deploying the application or while defining an application process step.

When deploying an application, you can specify parameter values that determine how to deploy the application. The parameter value can be a credential that specifies who can deploy the application, a value that represents the specific artifacts and artifact versions to deploy, the name of a specific schedule to apply, or the name of a snapshot to deploy. You can also specify if the deployment type is full deploy, smart deploy, or partial deploy. You can specify parameter values the same way in a process step to determine when or how to execute it.

How to get here: One of these ways:

- From the Home page, click the **Applications** launch pad, click the application name, select an application process, click the **Menu** button in the Application Process Visual Editor, and click **Parameters**.
- From the main menu on the Home page, click **Applications**, click the application name, select an application process, click the **Menu** button in the Application Process Visual Editor, and click **Parameters**.

The **Parameters** dialog box opens.



1	Number of parameters.
2	Name of the application process.
3	Name of the parameter.
4	Type of parameter: <ul style="list-style-type: none"> • Text Entry • Text Area • Dropdown Menu • Radio Selector • Checkbox • Credentials
5	This parameter is required when there is a check in this column.
6	Click this button to add a parameter.

When parameters are defined for the application process, a parameter list appears in the **Parameters** dialog box.

6 Parameters

Deploy

Select All

Name	Type	Required	
1. User Name	Text Entry	✓	>
2. Script/Command	Text Area		>
3. Stage	Dropdown menu	✓	>
4. Priority	Radio Selector	✓	>
5. QA Verification Required	Checkbox		>
6. User Name and Password	Credential	✓	>

1	<p>You can change the order in which the parameters appear in the UI form.</p> <div><div>↕</div><p>In desktop mode, mouse over the button and drag it to the new location.</p><div><div>↕</div><p>In touch screen mode, touch and hold the button and drag it to the new location.</p></div></div>
2	<p>Select a row and click the arrow at the end of the row to open the New Parameters dialog box.</p>

When you click **There are no Parameters yet. Add one +**, the **New Parameters** dialog box opens.

Parameters

New Parameter

1

Name

2

Description

3

Label

4

Select Parameter

5

☒ Required

6

☐ Defer Expansion

Cancel

OK

1	Name of the new parameter.
---	----------------------------

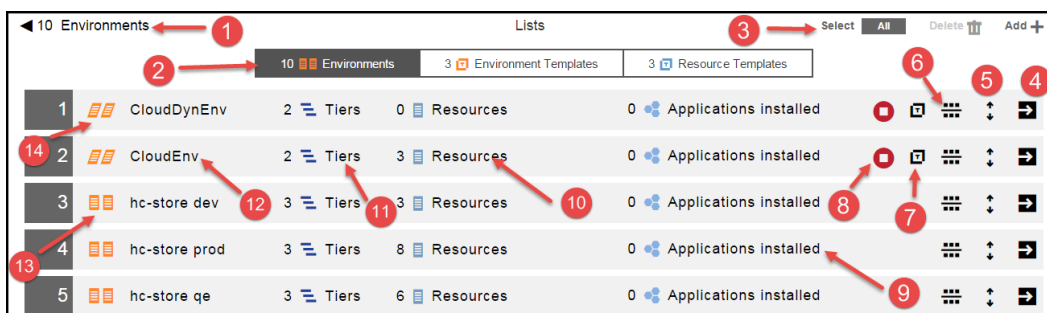
2	(Optional) Description of the new parameter.
3	<p>(Optional) Label for the parameter.</p> <p>If you enter a Label, it appears in the UI form when you deploy the application.</p> <p>If you do not enter a label, the parameter name appears in the UI form when you deploy the application.</p>
4	<p>Click the down arrow in the Select Parameter field to select the parameter type:</p> <ul style="list-style-type: none">• Text Entry• Text Area• Dropdown Menu• Radio Selector• Checkbox• Credentials <p>Depending on the type that you select, other fields appear in the dialog box.</p> <p>For examples, go to Setting Parameters for Application Processes on page 218.</p>
5	<p>Required check box</p> <p>When you click the Help button, this Help text appears:</p> <p>By default, this parameter is required. The application does not run when the parameter does not have a value.</p>
6	<p>Defer Expansion check box</p> <p>When you click the Help button, this Help text appears:</p> <p>When the Defer Expansion check box is selected and the parameter value contains \$[], ElectricFlow does not interpret it as a parameter reference and instead interprets it literally.</p>

Environments List

How to get here: One of these ways:

- From the Home page, click the **Environments** launch pad.
- From the main menu on the Home page, click the **Environments** destination.







The Environments List opens.



1	Breadcrumb showing the total number of environments in your ElectricFlow system.
2	<p>You can view the environments, environment templates, or resource templates for all of the environments in your system.</p> <ul style="list-style-type: none"> • Select the Environments tab to view the list of static, dynamic, and mixed environments. You can view details about the environments, environment tiers, and resources in the tiers. • Select the Environment Templates tab to view the list of available environment templates. You can use these templates to create environment when deploying an application. • Select the Resource Templates tab to view the list of available resource templates. Resource templates have the necessary information that ElectricFlow uses to dynamically spin up resources in the cloud when deploying an application. <p>By default, no environments, environment templates, and resource templates are configured. You create the ones you need to deploy applications in ElectricFlow.</p>
3	<p>Click the Select button to select all or none of the items in the list.</p> <p>Click the Delete button to delete the selected items.</p> <p>Click the Add + button to add an environment.</p>
4	Click the View details button to go to the Environments Visual Editor.
5	Click the View button to view more details about the environment.
6	Click the Inventory button to open the environment inventory, where you can see the status of an application as it runs and additional information about the application for troubleshooting.
7	Click the Template button to view the history of the environment template.

8	Click the Tear down button to retire the dynamic environment. All cloud resources will be deprovisioned (terminated). This button works only with dynamic environments.
9	Number of applications that run in the environment.
10	Number of resources in the environment.
11	Number of environment tiers in the environment.
12	Click the environment name (in this example, CloudEnv) to go to the Environments Visual Editor.
13	Static environment.
14	Dynamic environment.

You can determine the type of environment and the status of the environment by its icon and color:

Type of Environment	Available	Not Available
Static (only static resources)		
Dynamic (only dynamic resources)		
Mixed (some tiers with dynamic resources and other tiers with static resources)		

You can set the status of an environment in the Environments Visual Editor.

When you click the **Template** button, you view the history of the template:

1	CloudDynEnv	2 Tiers	0 Resources	TemplateName: AWSIest Tier: Tier 1 Creator: admin Used: 2x Created: 4/1/15
2	CloudEnv	2 Tiers	3 Resources	TemplateName: AWSIest Tier: Tier 2 Creator: admin Used: x Created: 4/1/15
3	hc-store dev	3 Tiers	3 Resources	
4	hc-store_prod	3 Tiers	8 Resources	

When you click the **View** button for a specific environment, details about the environment tiers and resources for the environment appear.

2	CloudEnv	2 Tiers	3 Resources	0 Applications Installed
1 Tier 1 /	CloudEnv-Tier_1	0 Resources		
2 Tier 2		3 Resources		
1	CloudEnv-Tier_2-re...	Modified: 04/01/15	By: admin	Live: 0 Days, 02:06 Hrs
2	CloudEnv-Tier_2-re...	Modified: 04/01/15	By: admin	Live: 0 Days, 02:06 Hrs
3	CloudEnv-Tier_2-re...	Modified: 04/01/15	By: admin	Live: 0 Days, 02:06 Hrs

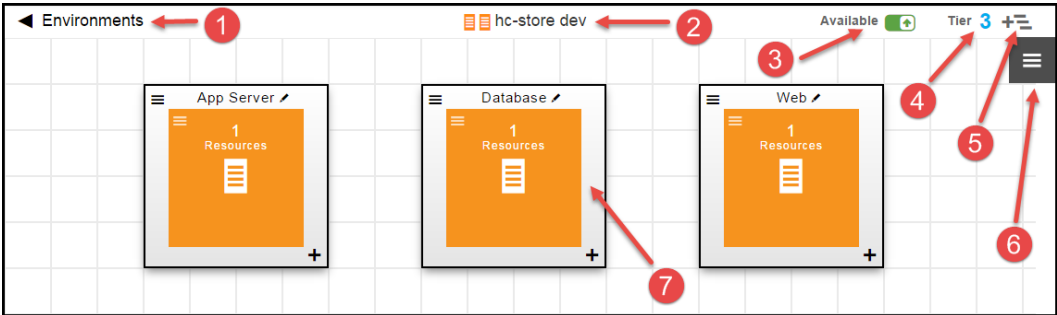
1	Name of the environment tier with the tier row numbering. Tier 1 is the first tier and Tier 2 is the second tier.
2	Name of the dynamic resource pool consisting of cloud resources.
3	Number of resources in the environment tier.
4	Click the Hide button to show less details. When you click the Hide button for an environment tier, the environment tier details disappear.
5	Click the View button to view more details about the object in the automation platform.
6	How long the resource has been available since it was created.
7	User who performed that last activity on the resource.
8	Last activity on the resource.
9	Name of the resource with the resource row number.

Environments Visual Editor

How to get to here: From the Environments List page, select an environment.

Environments consist of resources that are grouped into tiers. The component and application processes run on resources assigned to environments.

To run applications, you must configure tier maps of application tiers to environment tiers.

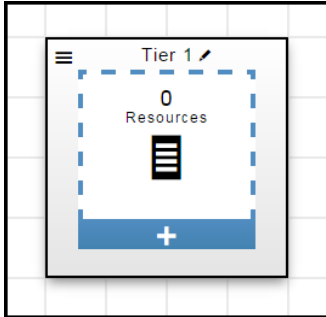


The following information is on this page:

1	Breadcrumb to the environment
2	Name of the environment
3	<div>Button showing the status of the environment<ul style="list-style-type: none">The environment is available when it is green.It is unavailable when it is red.To change the status from Available to Unavailable, click this button. It becomes<div><div>Unavailable</div><div></div></div>To change the status to Available, click the button again.</div>
4	Number of environment tiers
5	Click the Add tier button to add an environment tier.
6	<div>Click the Menu button to to open the context menu. You can get more details about the environment:<ul style="list-style-type: none">Details–The name and description of the object.Properties–The properties in the object.Access Control–The access control configuration in the automation platform for the object.Track Changes–The change history of the object.Delete–Delete this object.</div>

The appearance of the resources in an environment tier changes when you assign a resource that is defined and managed by the automation platform.

After you add an environment tier:



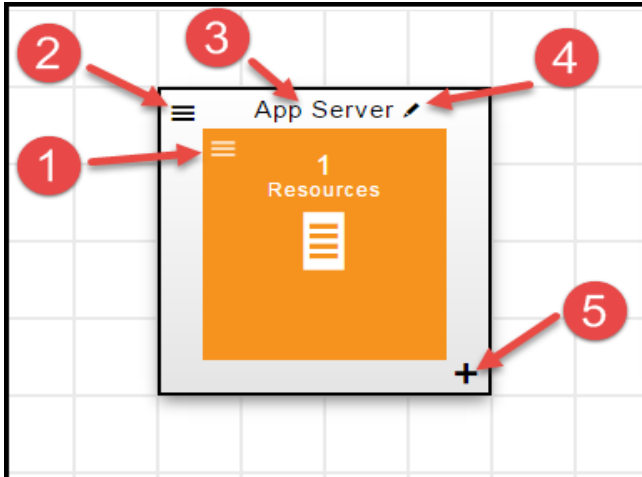
After you assign a resource to the tier:



Environment Tiers

How to get here: From the Environments Visual Editor, choose an environment tier.

Environment tiers consist of resources that can be assigned to applications.

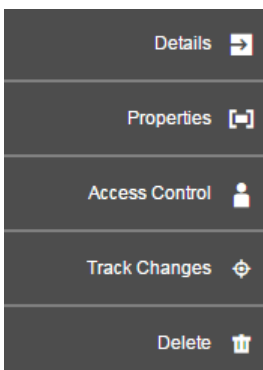


The following information is available about the tier.

1	Click the Menu button to view the resource details.
2	Click the Menu button to view the tier details.
3	Name of the environment tier
4	Click the Edit button to edit the tier name and description in the Environment Tier Details dialog box.
5	Click the + button to add a resource to the environment tier.

Context Menu for the Environment Tier

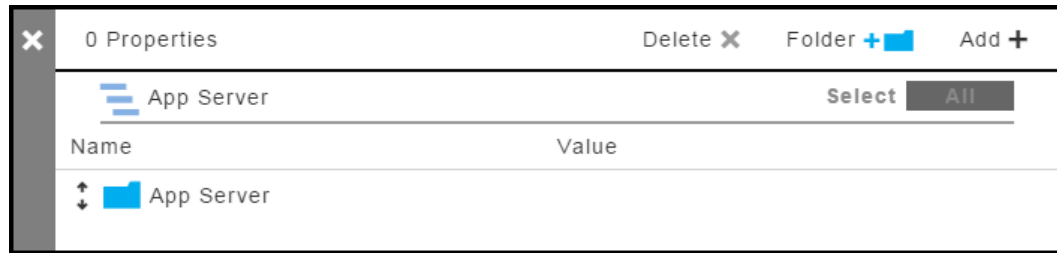
This menu appears when you click the **Menu** button:



- When you click **Details**, the **Environment Tier Details** dialog box appears.

You can edit the environment name and description.

- When you click **Properties**, the Properties dialog box appears, where you can set the properties for the environment tier.



- When you click **Access Control**, you go to the Access Control page for the tier in the automation platform, where you can set privileges for the objects in your environment.

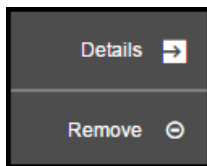
For more information about the Access Control page, go to the automation platform Help > **Projects** > **Select a project** > **Access Control**.

- When you click **Track Changes**, the Change History Search Form opens.
- When you click **Delete**, the **Delete Environment Tier** dialog box appears.

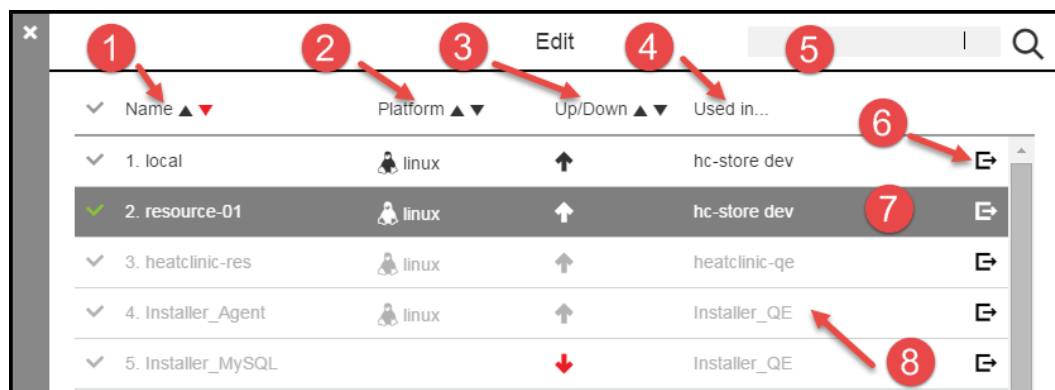
Click **OK** to verify that you want to delete this environment tier.

Context Menu for the Resource

This menu appears when you click the **Menu** button for a resource.



- When you click **Details**, the resource list appears. It shows the resource name, the resource platform, the status (up or down), and where it is used.



1	Name of the resource
2	The resource platform
3	Status of the resource: Up or down.
4	Environment to which the resource is assigned
5	Enter the search criteria in this field to search for specific resources.
6	Click the View details button to go to the Resources page in the automation platform. For more information about the Resources page, go to the automation platform Help > Web Interface Help > Resources .
7	Details about the resource you selected. If you want to change the resource details, you can replace this resource with an available resource.
8	This resource is not available because it is assigned to an environment other than the <i>hc-store dev</i> environment.

- When you click **Remove**, the resource list appears.

After selecting the resources that you want to delete, click **OK**.

The resources that you selected are deleted from the environment tier but are not deleted from the application or the automation platform.

Adding Resources to an Environment Tier

In an environment tier, click the **+** button to open the resource list, select one or more available resources in the resource list, and click **OK**.

The environment tier in the Environment Visual Editor is updated and now shows that there are more resources in the environment tier.

Environment Templates List

How to get here: One of these ways:

- From the Home page, click the **Environments** launch pad, and then click the **Environment Templates** tab.
- From the main menu on the Home page, click **Environments > Environment Templates**.

The Environment Templates List opens.

10 Environments						
Lists						
<div> <div>10 Environments</div> <div>3 Environment Templates</div> <div>3 Resource Templates</div> </div>						
1	AWStest	3 Tiers	Modified: 04/01/15	By: admin	1 Live	Used: 2
2	ET1	1 Tiers	Modified: 04/01/15	By: admin	0 Live	Used: 0
3	OSTest	2 Tiers	Modified: 04/01/15	By: admin	0 Live	Used: 0

1	Name of the environment template.
2	Number of tiers in the template.
3	Last activity on the environment template.
4	User who performed that last activity on the resource.
5	Number of environments currently live from this template.
6	Number of times that this template has been used.
7	Click the Track Changes button to start a change history search.
8	Click the View details button to go to the Environment Templates Visual Editor. It has the same elements as the Environments Visual Editor.

This is the Environment Templates Visual Editor for *AWStest*.

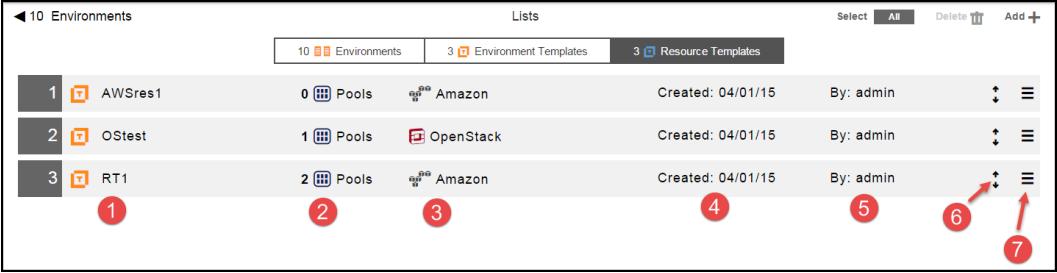
Environment Templates		AWStest		Tier 3
Tier 1	16 Resources	Tier 2	3 Resources	Tier 3

Resource Templates List

How to get here: One of these ways:

- From the Home page, click the **Environments** launch pad, and then click the **Resource Templates** tab.
- From the main menu on the Home page, click **Environments > Resource Templates**.

The Resource Templates List opens.



1	Name of the resource template.
2	Number of resource pools.
3	Third-party plugin for the cloud provider or configuration management.
4	Last activity on the environment template.
5	User who performed that last activity on the resource.
6	Click the View button to view more information about the resource template.
7	Click the Menu button to view details about the resource template.
8	Click the View details button to view more details about the resource pool.
9	How long the resource has been available since it was created.
10	Name of the resource pool.

Tier Maps

How to see a Tier Map: From the Applications Designer, click the down arrow in this button to show a drop-down list of configured tier maps. Then select a tier map to view.

Example:



How to add a Tier Map: Click the **Add tier map** button to add a tier map an application.

Example:



A tier map is a mapping of application tiers to environment tiers for a specific application and environment.

You must configure a tier map if you want to run an application.

This tier map shows the mapping between the Heat Clinic Store 1.1 application and the environment called hc-store dev. Each application tier is mapped to an environment.

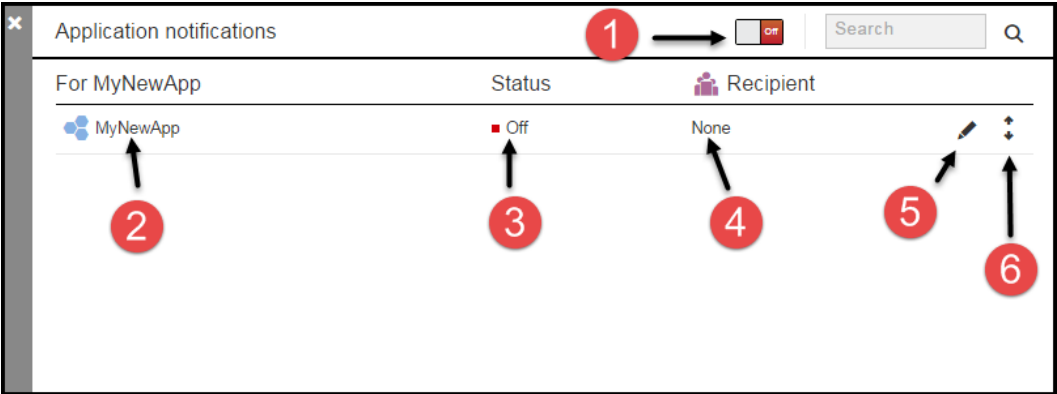
Application Tiers		Environment Tiers
Tomcat	✓	App Server
MySQL	✓	Database
Apache Web	✓	Web

You can map more than one application tier to the same environment tier.

Application Tiers		Environment Tiers
Tomcat	✓	Tomcat
MySQL	✓	Apache Web
Apache Web	✓	Apache Web

Application Notifications Dialog Box

How to get here: From the Applications Visual Editor, click the **Menu** button, and select **Notifications**.



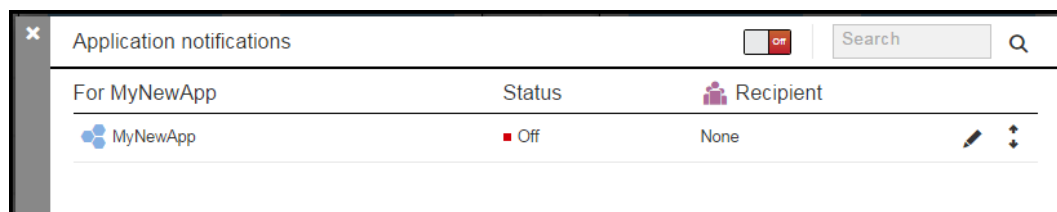
The following information is in this dialog box.

1	<p>Notifications toggle.</p> <p>Click this to enable (On) or disable (Off) email notifications for the application.</p> <ul style="list-style-type: none">• Enable–The system sends email notifications to the specified recipients.• Disable–The system do not send email notifications.
2	<p>Name of the application.</p>
3	<p>Status of the notification for the object at the current level.</p> <ul style="list-style-type: none">• On–The system sends email notifications.• Off–The system do not send email notifications.
4	<p>How many recipients receive notifications.</p> <p>The values are None and Some (one or more).</p> <p>When Some is in the Recipient column, click the Edit button to see who the recipients are</p>
5	<p>Edit button.</p> <p>Click this to edit a notification. The "Application notifications / edit" dialog box opens.</p>

6	<p>View button.</p> <p>Click this to view the notifications for the object one level below the current level.</p> <p>In this example, when you click this button in the MyNewApp row, the application processes in the MyCoolApp application appear in the rows below.</p>
---	---

Setting Notifications for the First Time

- New email notifications are disabled in the application, its application processes, and the application process steps (type: process) before you configure them.



- You configure notifications in the "Application notifications / edit" dialog box.

IMPORTANT: The first time that you set notifications in this dialog box, the Notifications toggle changes to **On**. After you enter notification settings and click **OK**, email notifications are enabled at that level.

To see the notification settings for the application processes and process steps, click the **View** button. The settings for the application processes appear.

Example:



Click the **View** button for each application process. The settings for processes steps appear.

Application notifications			
		<input type="checkbox"/> Off	<input type="text" value="Search"/>
For MyNewApp	Status	Recipient	
MyNewApp	Off	None	
1 New installation	Off	None	
1 install new	Off	None	
2 prepare the system	Off	None	
2 existing installation	Off	None	
1 backup	Off	None	
2 set new database	Off	None	
3 upgrade	Off	None	

Enabling Notifications

You can enable notifications at the application, application process, and process levels.

To enable email notifications at the application level:

- Click the Notifications toggle and change it to **On**.

The status of the application changes to **On**.

Example:

Application notifications			
		<input checked="" type="checkbox"/> On	<input type="text" value="Search"/>
For MyNewApp	Status	Recipient	
MyNewApp	On	None	
1 New installation	Off	None	
2 existing installation	Off	None	

- Click the **Edit** button to open the **Application notifications / edit** dialog box.

The **Application notifications / edit** dialog box appears. The Notification toggle changes to **On**.

Example:

Application notifications / edit

MyNewApp On

Who When Where +

Add users, groups, or email addresses: + Event: Environment:

| Both Failed and Success... All -

To enable notifications at the application process and process step levels, go to the **Application Notifications / edit** dialog box for the specific process or process step.

The dialog box opens, and the Notifications toggle is now **On**.

Example:

Application notifications / edit

New installation On

Who When Where +

Add users, groups, or email addresses: + Event: Environment:

Both Failed and Success... All -

Cancel OK

When you enter notification settings in the dialog box and click **OK**, the settings are saved. The **Application notifications** dialog box appears and now shows that the application process status is **On**.

Example:

Application notifications			
For MyNewApp		Status	Recipient
MyNewApp		On	Some
1	New installation	On	Some
2	existing installation	Off	None

Adding Notifications

You can configure one or more notifications in an application process or other object.

Configuring recipients

In the **Who** field, you add users or groups who are configured and managed in the ElectricFlow platform or email addresses.

When you start typing a user name, group name, or email addresses, a list of names or email addresses appear that match what you are typing.

Example:

Application notifications / edit

backup

On

Who

When

Where

Add users, groups, or email addresses:

Event:

Environment:

a|

Both Failed and Succes...

All

admin (?)

admin-asia (?)

admin-aus (?)

admin-uk (?)

admin-us (?)

jadams (?)

sclaus (?)

If one of the suggestions matches the name or email address, select it, or continue typing. You can add more than one name or email address.

Example:

Application notifications / edit

backup On

Who When Where +

Add users, groups, or email addresses:

admin admin-asia sclaus jadams userX@gmail.com

DevT200@gmail.com

Event: Both Failed and Success... ▼

Environment: All [-]

Configuring the event that triggers the notification

In the **When** field, you select the event that triggers a notification to be sent to the recipients in the **Who** field. The default is **Both Failed and Successful**. Click in the **When** field to select the event for the notification.

Example:

Application notifications / edit

backup On

Who When Where +

Add users, groups, or email addresses:

admin admin-asia sclaus jadams userX@gmail.com

DevT200@gmail.com

Event: Both Failed and Success... ▼

Run Failed

Run Successful

Both Failed and Successful

Environment: All [-]

Configuring the environments where the notification applies

In the **Where** field, you select the environments to which the notifications apply. Click in the **Where** field to select the environments, which are the environments to which the application is mapped in the tier map.

Example:

Application notifications / edit

backup On

Who	When	Where
Add users, groups, or email addresses: + admin admin-asia schlaus jadams userX@gmail.com DevT200@gmail.com	Event: Run Successful ▼	Environment: All hc-store dev

Adding More Notifications

Click the **Add Notifications** button to add a new notification.

Example:



After you have added your email notifications, click **OK** to save the settings and return to the Application notifications dialog box.

Example:

Application notifications / edit

backup On

Who	When	Where
Add users, groups, or email addresses: + jadams admin-asia schlaus admin userX@gmail.com DevT200@gmail.com	Event: Run Successful ▼	Environment: hc-store dev
admin-uk admin-aus DevLead@electric-cloud.com UserAZ@gmail.com	Event: Both Failed and Succes... ▼	Environment: All

Deleting Notifications

Click the **Delete Notification** button to delete a notification when there are more than two notifications.



The screenshot shows a dialog box titled "Application notifications / edit" for an application named "MyApp". At the top right, there is a toggle switch labeled "On". Below this, the dialog is organized into three columns: "Who", "When", and "Where".

- Who:** A section labeled "Add users, groups, or email addresses:" containing a list of users: "ajones", "haaron@electric-cloud.com", "admin", "admin2", "DevT2000", and "bbanks@electric-cloud.com".
- When:** A section labeled "Event:" with a dropdown menu currently showing "Both Failed and Succes...".
- Where:** A section labeled "Environment:" with a dropdown menu currently showing "hc-store dev".

At the bottom of the dialog, there are "Cancel" and "OK" buttons.

Disabling Notifications

You can disable all email notifications or specific ones in the application.

- When you disable the email notifications for an application, all email notifications, including the ones for application processes and process steps, are disabled.
- You can disable the email notifications for an application process but still keep the notifications enabled for process steps in the application.

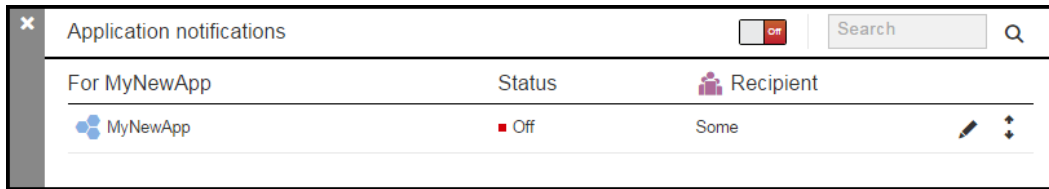
Applications

To disable email notifications for the application:

- Click the Notifications toggle in Application notifications dialog box.

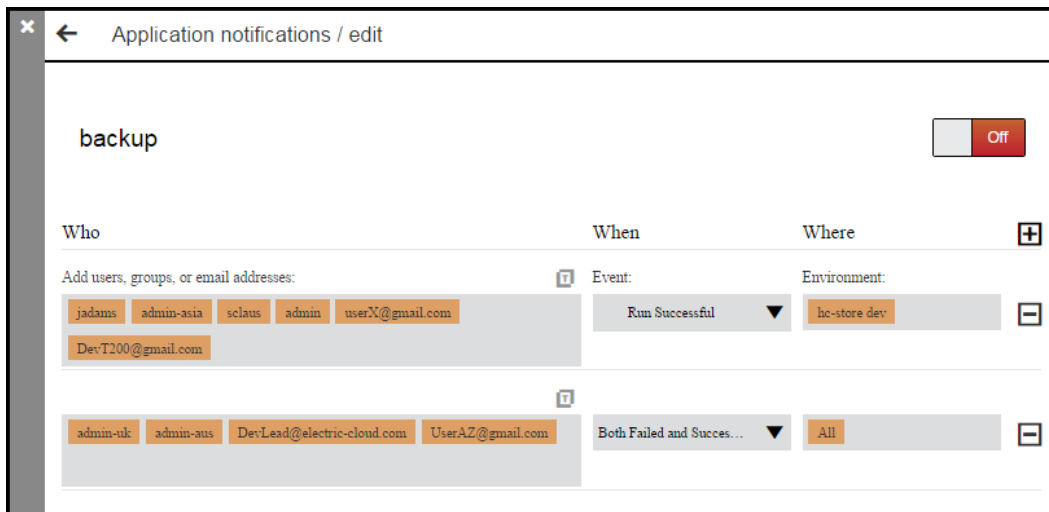
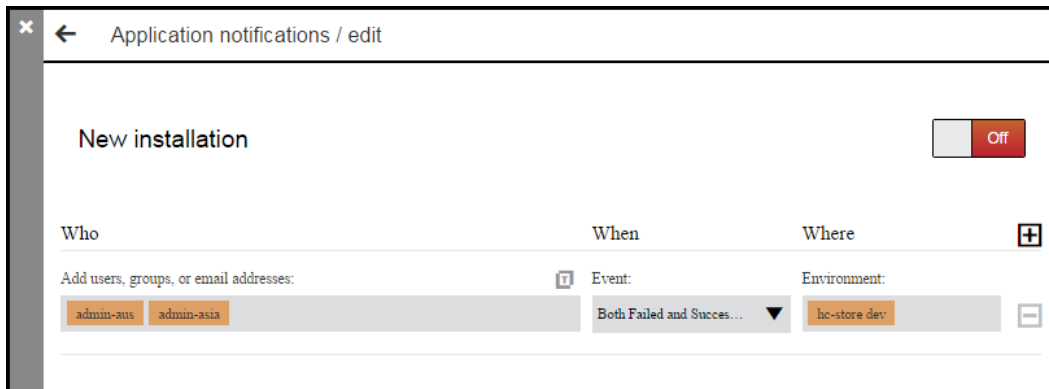
The toggle and the status of the application change to **Off**.

- When you use the "Applications notification / edit" dialog box:
 - a. Click the **Edit** button to open the dialog box.
 - b. Click the toggle to change it to **Off**, and then click **OK**. The Application notifications dialog box opens with the updated status.



Application Processes and Process Steps

To disable notifications, click the Notifications toggle and then click **OK** in "Application notifications / edit" dialog box. The toggle changes to **Off**.



The Application notifications dialog box now shows that email notifications are disabled for the application, its application processes, and process steps.

Application notifications

Off

Search

For MyNewApp	Status	Recipient	
<div>MyNewApp</div>	Off	Some	<div></div> <div></div>
1 <div>New installation</div>	Off	Some	<div></div> <div></div>
1 <div>install new</div>	Off	None	<div></div>
2 <div>prepare the system</div>	Off	None	<div></div>
2 <div>existing installation</div>	Off	None	<div></div> <div></div>
1 <div>backup</div>	Off	Some	<div></div>
2 <div>set new database</div>	Off	None	<div></div>
3 <div>upgrade</div>	Off	None	<div></div>

Change History Search Form

How to get here: From most pages, click the **Search** button to open the "Change History - Search" form.

Example:



The "Change History - Search" form has the following information:

1

2

Change History - Search

Last changes

All...

3

1	<p>Time range field.</p> <p>Click the down arrow to open the drop-down list of start times.</p> <p>The end time is the current time.</p>
2	<p>Objects field.</p> <p>Click the down arrow to open the drop-down list of objects to include in the search. You can select All or specific objects.</p> <p>By default, seven of the most commonly tracked objects are selected.</p>

3	<p>Search criteria.</p> <p>After you type, the system starts searching for objects based on the time range and objects that you selected.</p> <p>The search results are in the Change History.</p>
---	--

Change History Page

How to get here: Click the **Change History** button for a tracked object.

Example:



The Change History has this information about the object called *Proc*:

1

Change History for Proc

2

3

12/04/14 - 10:45 AM

10

12/04/14 - 7:05 PM

▼ Last Successful Run

View All Changes

Objects

Process (2)

Process Step (3)

Process Dependency (1)

Ad (4)

Property Sheet (3)

Property (3)

Changes

Modified (4)

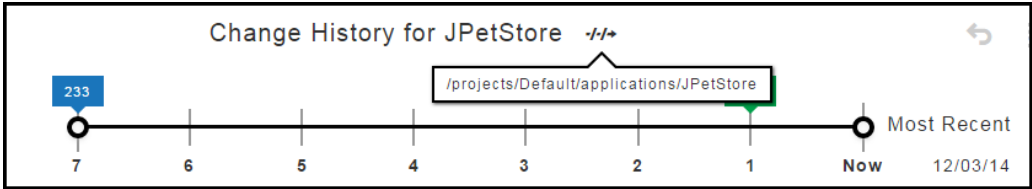


Created (12)

Changed by...

Project: Default (1)

Admin (15)

When	What	Name	By...	Change	Path
1 Dec 04, 2014 11:08 AM Pacif...	process	proc	project: Default	modified	
2 Dec 04, 2014 11:08 AM Pacif...	processStep	s2	admin	modified	
3 Dec 04, 2014 11:07 AM Pacif...	processDepend...	d086ed98-7be8...	admin	created	
4 Dec 04, 2014 11:07 AM Pacif...	acl	ec_deploy	admin	created	
5 Dec 04, 2014 11:07 AM Pacif...	acl	d3	admin	created	
6 Dec 04, 2014 11:07 AM Pacif...	propertySheet	d3	admin	created	
7 Dec 04, 2014 11:07 AM Pacif...	propertySheet	ec_deploy	admin	created	
8 Dec 04, 2014 11:07 AM Pacif...	propertySheet	d3	admin	created	

1	<p>Time line</p> <p>You can modify the start and end times.</p> <p>Default:</p> <ul style="list-style-type: none"> • The entire time line is selected. All changes appear in the Change History list. • The time increment is from the Last Successful Run to the most current change. • The start time is based when the last successful run occurred. • The end time is when the most recent changed occurred.
2	<p>Path to the tracked object.</p> <p>Example:</p>  <p>The diagram shows a horizontal timeline titled "Change History for JPetStore". On the left, a blue box labeled "233" is positioned above a tick mark labeled "7". The timeline has tick marks labeled 7, 6, 5, 4, 3, 2, 1, and "Now". A green vertical bar is at the "1" mark. A callout box points to the timeline with the path "/projects/Default/applications/JPetStore". On the right, there is a "Most Recent" label and a date "12/03/14". A circular arrow icon is in the top right corner of the diagram area.</p>
3	 <p>Click  to revert the selected changes.</p>
6	<p>Time line.</p> <p>The start time is based on the time range that you selected.</p> <p>The end time is the current time.</p> <p>You can manually change the start and end times after you run the search and get the search results.</p>
4	<p>Filters for the change history.</p> <p>You can view all changes or view only selected changes.</p> <p>The objects in the list are the objects in the change history search results.</p>

5

Change history for the selected object.

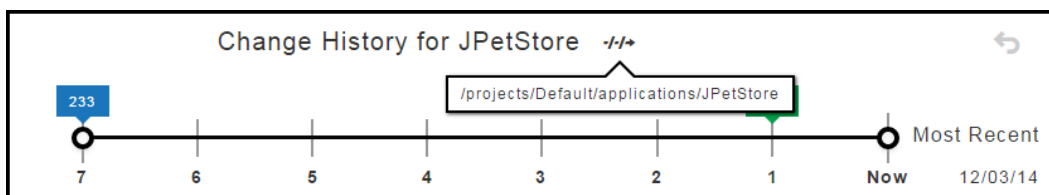
- When—the date and time that the object changed.
- What—The type of object.
- Name—The name of the object.
- By—The "user" that changed the object, which can be a project or a user.
- Change—The type of change.
- Path—Click the **View Path** button to see the path to the object.

	When ▲ ▼	What	Name	By...	Change	Path
1	Dec 03, 2014 4:34 PM Pacific...	property	jobcounter	project: ...	modified	⚙️
2	Dec 03, 2014 4:34 PM Pacific...	process	Before path: /projects/Default/applications/JPetStore/jobcounter After path: /projects/Default/applications/JPetStore/jobcounter			⚙️
3	Dec 03, 2014 12:38 PM Pacific...	property	pluginpr...	admin	created	⚙️

Click the **View** button to view more information about a specific change in the change history.

Paths to Objects

Click the **View Path** button next to the "Change History for JPetStore" title to see the path to the application.

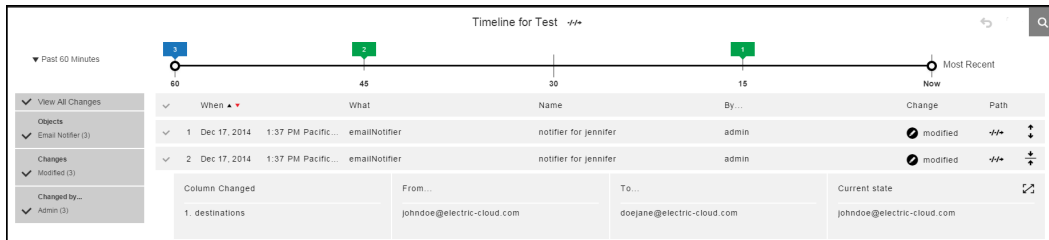


Click the **View Path** button to see the change in the path to the object before and after the change.

	When ▲ ▼	What	Name	By...	Change	Path
1	Dec 03, 2014 4:34 PM Pacific...	property	jobcounter	project: ...	modified	⚙️
2	Dec 03, 2014 4:34 PM Pacific...	process	Before path: /projects/Default/applications/JPetStore/jobcounter After path: /projects/Default/applications/JPetStore/jobcounter			⚙️
3	Dec 03, 2014 12:38 PM Pacific...	property	pluginpr...	admin	created	⚙️

Detailed Object Changes

Click the **View** button to see the change in the property called emailNotifier.



Click the **Expand** button to all the changes to the property,

Example:



When you click the **Expand** button in a cell, you can see more details in the current change in the cell.

If you click the **Select All** button, all the changes about the object appear.

X ← 2 Dec 17, 2014 1:37 PM Pacific Standard Time emailNotifier			
notifier for jennifer By: admin modified			
Column Changed	From...	To...	Current state
1. destinations	doejane@electric-cloud.com	doejane@electric-cloud.com	john.doe@electric-cloud.com

To select an object to revert or import the changes to an XML file, select the row of the object in the expanded view.

✓ 2 Dec 17, 2014 1:37 PM Pacific... emailNotifier			
notifier for jennifer admin modified			
Column Changed	From...	To...	Current state
1. destinations	john.doe@electric-cloud.com	doejane@electric-cloud.com	john.doe@electric-cloud.com

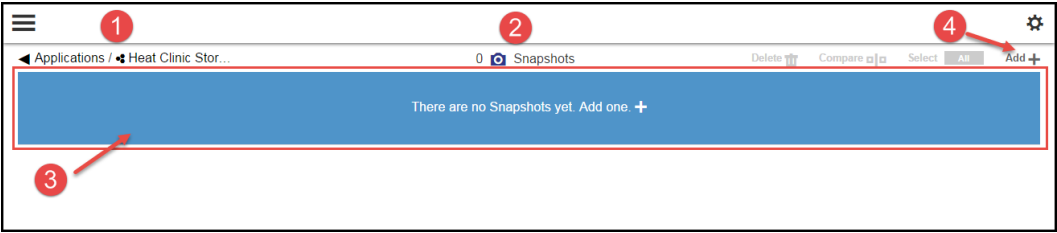
Snapshot List

How to get here: From the Applications List, select an application > click the **Snapshot** button > select **Snapshot List**.

Example:

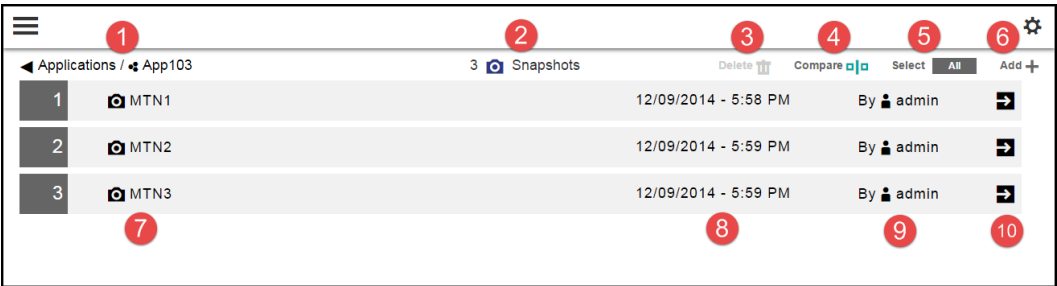


This example shows a Snapshot List with no snapshots:



1	Breadcrumb to the Snapshot List: <i>Applications/<Application name></i>
2	Number of snapshots.
3	Add + pane. Click anywhere in the pane to add a snapshot. The New Snapshot dialog box opens.
4	Click on Add + to add a snapshot. The New Snapshot dialog box opens.

This Snapshot List has three snapshots:



1	Breadcrumb to the Snapshot List: <i>Applications/<Application name></i>
2	Number of snapshots.
3	Delete button. After you select one or more snapshots in the list, the Delete button is available (enabled). Click this to delete the snapshots that you selected.

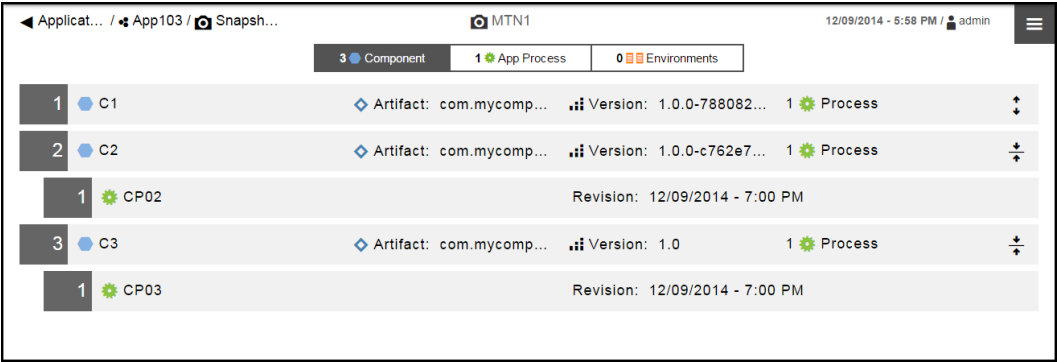
4	<p>Compare snapshots from a list.</p> <p>This is available (enabled) when the list has two or more snapshots.</p> <p>Click this to open a full-screen window and view two snapshots next to each other.</p> <p>After the window opens, you can select snapshots on both sides and compare those snapshots.</p> <p>The default is have the most recent snapshot on the left and the previous snapshot on the right.</p> <p>You do not have to select a snapshot to open the full-screen window and view two snapshots.</p>
5	<p>If you click All, all the snapshots are selected.</p> <p>If you click None, none of the snapshots are selected.</p>
6	Add a snapshot.
7	Snapshot name.
8	Time stamp when the snapshot was created.
9	The user who created the snapshot.
10	<p>View details button.</p> <p>Click this to go to the Snapshot Details pages.</p>

Snapshot Details Page

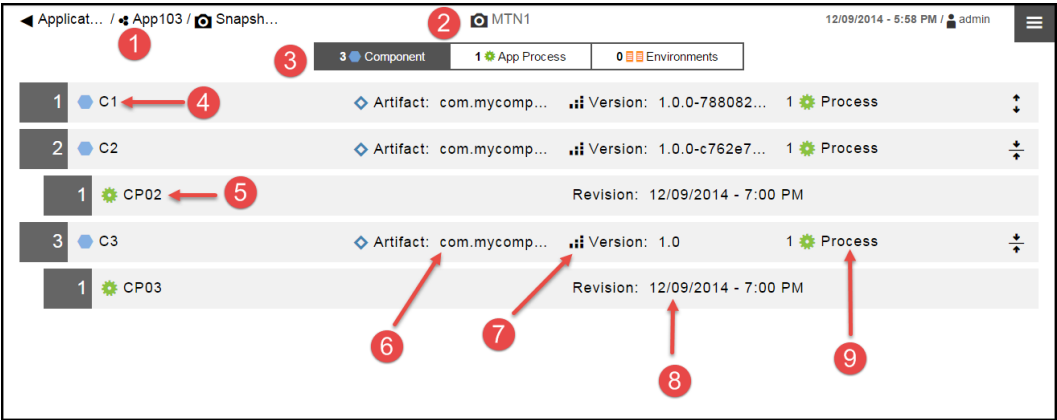
How to get here: From the Snapshot List, click the **View Details** button.

Example:



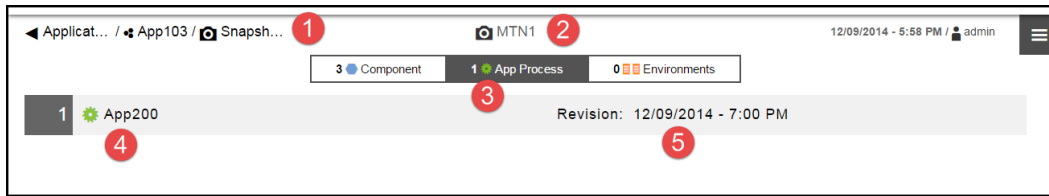


This is the Component view.



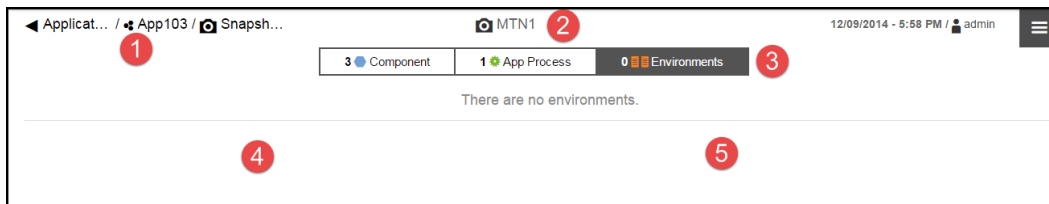
1	Breadcrumb to this Snapshot Details page: <i>Applications/<Application name>/Snapshots</i>
2	Name of the snapshot.
3	Number of components in the snapshot.
4	Component name.
5	Component process name.
6	Artifact name.
7	Artifact version.
8	Time stamp when the component process was last modified.
9	Number of component process for the component.

This is the App Process view.



1	Breadcrumb to this Snapshot Details page: <i>Applications/<Application name>/Snapshots</i>
2	Name of the snapshot.
3	Number of application processes in the snapshot.
4	Application process name.
5	Time stamp when the application process was last modified.

This is the Environment view.



1	Breadcrumb to this Snapshot Details page: <i>Applications/<Application name>/Snapshots</i>
2	Name of the snapshot.
3	Number of environments in the snapshot.
4	Environment name.
5	Time stamp when the environment was last modified.

Snapshot Dialog Boxes

[New Snapshot Dialog Box](#) on page 410

[Component View in the New/Preview Dialog Box](#) on page 410

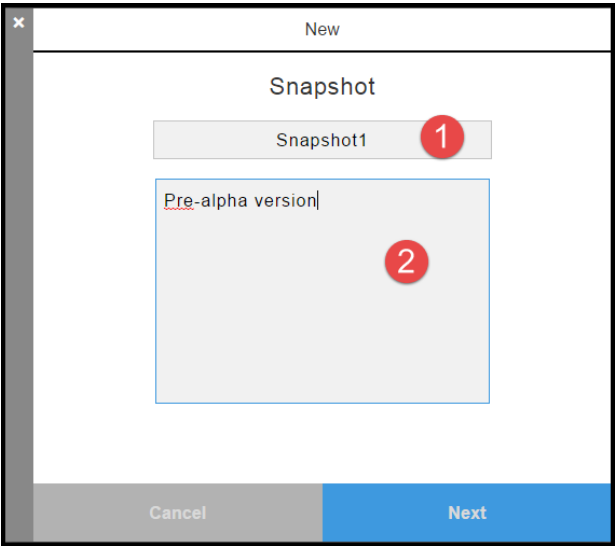
[Modifying the Artifact Version](#) on page 413

[App Process View in the New/Preview Dialog Box](#) on page 413

New Snapshot Dialog Box

How to get here: From the Applications List, click the **Snapshot** button > select **Snapshot List** > click in the **Add one. +** pane or click **Add+** in the Snapshot List.

The New Snapshot dialog box opens.

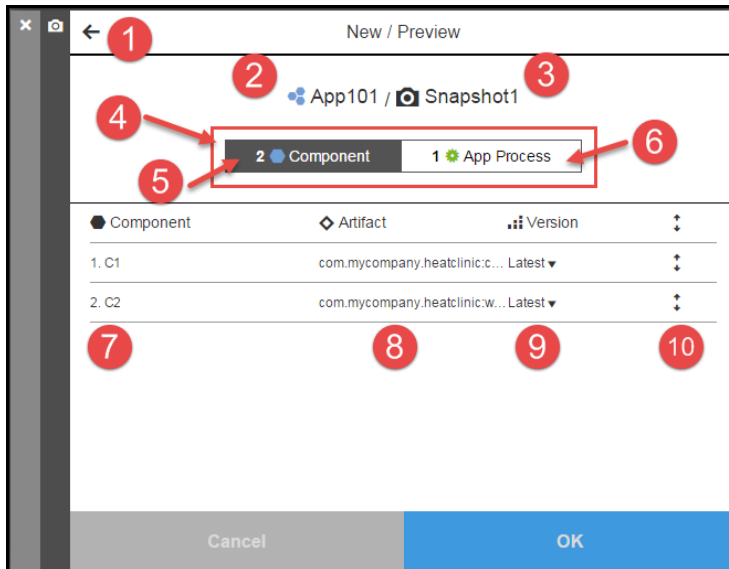


1	Name for the snapshot that must be unique within the application.
2	(Optional) Description of the snapshot.

Component View in the New/Preview Dialog Box

How to get here: In the New Snapshot dialog box, enter the snapshot name and the optional description, and click **Next**.

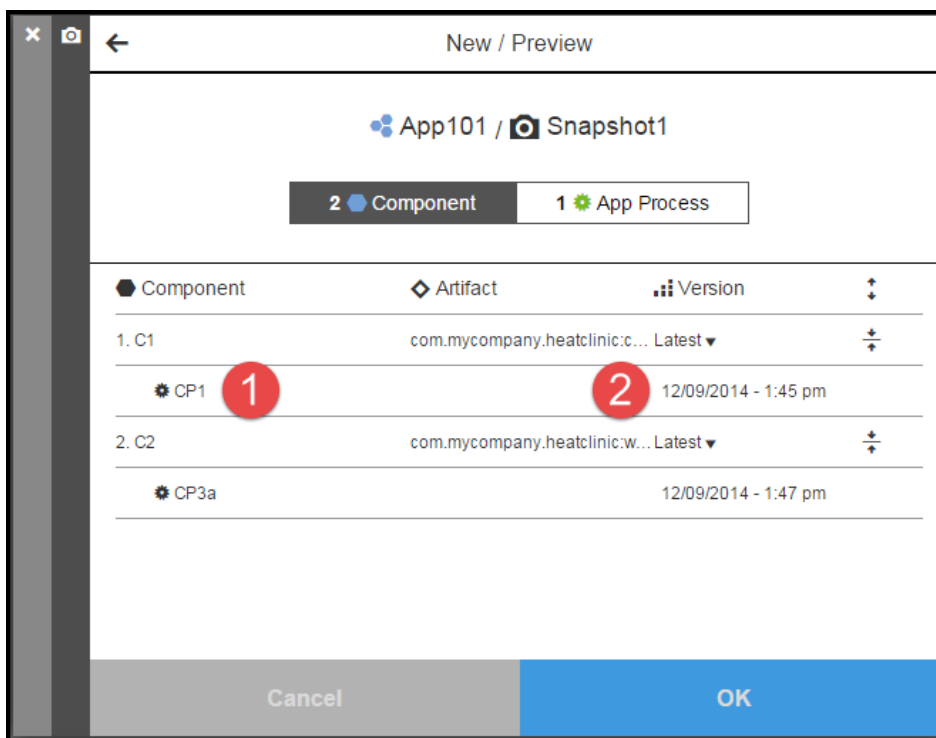
The New/Preview dialog box opens.



1	Go back to the New Snapshot dialog box.
2	Application name.
3	Snapshot name.
4	<p>Toggle between the Component, App Process, and Environment views. Before a snapshot is deployed, only the Component and App Process views are available.</p> <div data-bbox="389 1184 893 1379"> </div> <p>After the snapshot is deployed, Component, App Process, and Environment views are available.</p> <div data-bbox="389 1551 1088 1829"> </div>

5	Number of components.
6	Number of application processes.
7	Component number (not the row number) and component name. When you click the View icon to show the component details, the component processes rows do not have numbered.
8	Name of the artifact associated with the component.
9	Version of the artifact. Click the down arrow to open the drop-down menu where you can select a different artifact version.
10	Click the View button to show the component details.

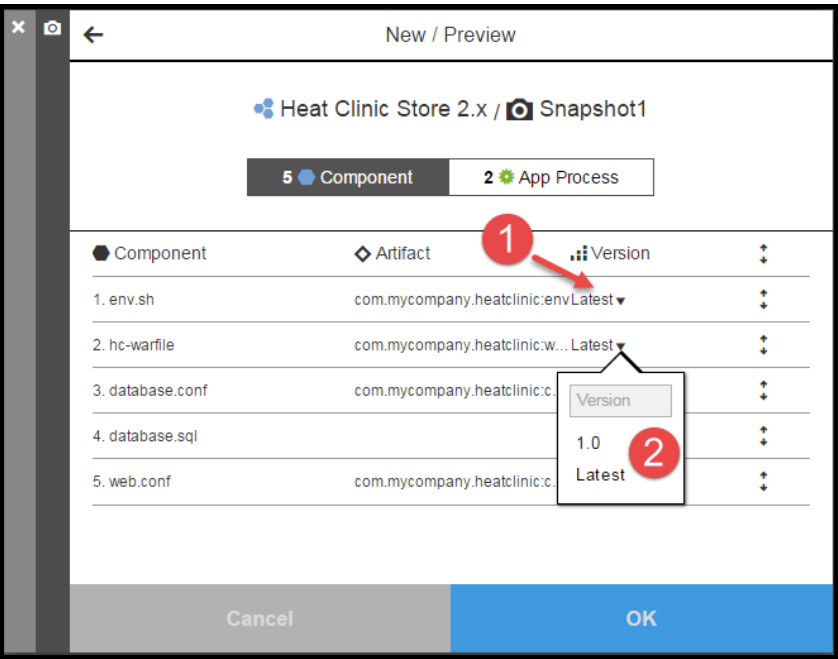
When you choose a component and click the **View** button, the component process information appears.



1	Name of the component process.
2	Time stamp when the component process was last run.

Modifying the Artifact Version

This example shows the current version of each component, which is Latest.



1	Current version of the artifact.
2	Possible artifact versions. To modify the artifact version in the snapshot, click the down arrow in the Version column to open a drop-down menu. You can select a version or enter the one that you want to use.

App Process View in the New/Preview Dialog Box

When you toggle to the App Process view, the following information appears.

1	Application name.
2	Snapshot name.
3	Number of application processes in the application.
4	Name of the application process.
5	Time stamp when the component process was last run.

Inventory Tracking

Electric Flow uses Inventory Tracking to track what is built, tested, and deployed in your continuous delivery solution, including artifacts, the artifact version, resources on which the applications are run, and environments to which the resources are assigned.

If there is an issue in an application in operations, you can find the details about what was deployed with Inventory Tracking.

Tracking at the Component Process Level

Inventory Tracking occurs at the component process level.

You design the component process as one of the following process types in the Component Process Details dialog box:

- **Deploy**—Enables Inventory Tracking. The ElectricFlow server tracks artifacts deployed to environments. This is the default.
- **Undeploy**—After the first successful job step in a component process with this setting, the automation

platform removes the environment inventory record.

- **Other**—Disables Inventory Tracking.

New

Component Process Details

Name

Description

Deploy

Credential

0

Optional >

Workspace

default

Optional

Time limit

0

Seconds

Optional

Cancel

OK

Application Inventory

How to get here:

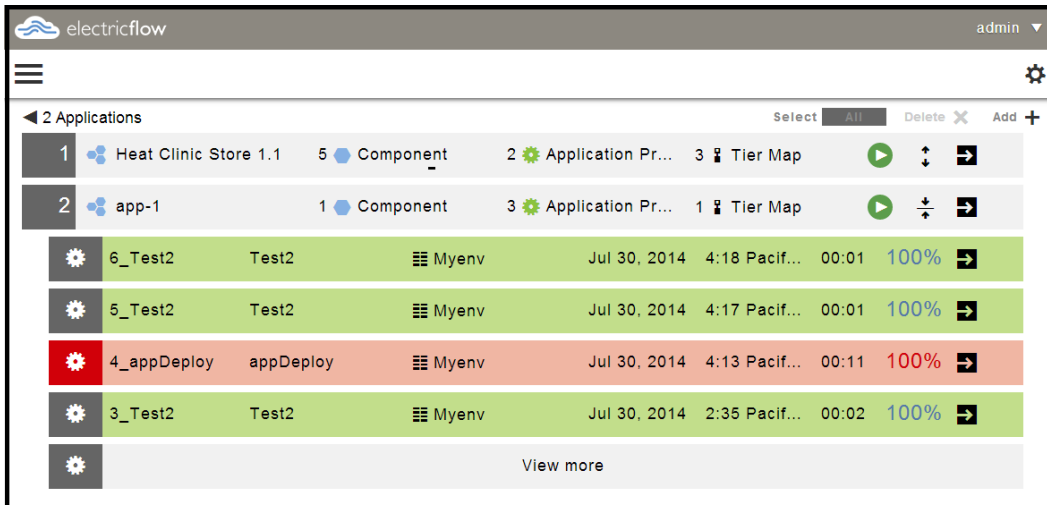
- From the Home page, click the **Applications** launch pad.
- From the main menu on the Home page, click the **Applications** destination.

The Application Inventory is the state of the application at a point in time.

- When an application is running, you can see the progress as it runs.
- After an application runs, you can see the results for any object in application.

Click the **View details** button in a row to see more details about object.

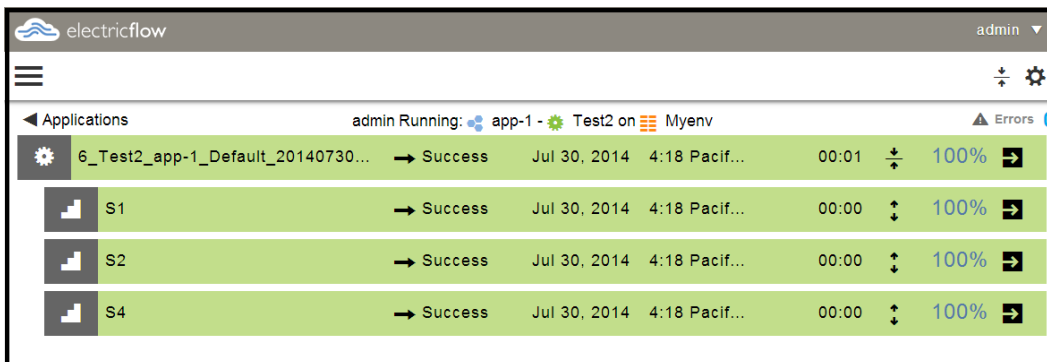
The following example shows that the application called "app-1" ran successfully on July 30, 2014, at 4:18 pm.



The screenshot shows the ElectricFlow dashboard with a header bar containing the logo, the name 'electricflow', and a user dropdown menu set to 'admin'. Below the header is a navigation menu with a hamburger icon and a settings gear icon. The main content area is titled '2 Applications' and includes a toolbar with 'Select', 'Delete', and 'Add' buttons. The table lists two applications: 'Heat Clinic Store 1.1' and 'app-1'. Below these, a list of test results is shown for 'app-1'. The row for '6_Test2' is highlighted in green, indicating a successful run. The row for '4_appDeploy' is highlighted in red, indicating a failed run. The row for '3_Test2' is highlighted in green, indicating a successful run. A 'View more' link is present at the bottom of the table.

Application	Component	Application Pr...	Tier Map	Test Name	Environment	Date	Time	Status	Progress	Action
1	Heat Clinic Store 1.1	5	Component	2	Application Pr...	3	Tier Map			
2	app-1	1	Component	3	Application Pr...	1	Tier Map			
	6_Test2	Test2	Myenv	Jul 30, 2014	4:18 Pacif...	00:01	100%			
	5_Test2	Test2	Myenv	Jul 30, 2014	4:17 Pacif...	00:01	100%			
	4_appDeploy	appDeploy	Myenv	Jul 30, 2014	4:13 Pacif...	00:11	100%			
	3_Test2	Test2	Myenv	Jul 30, 2014	2:35 Pacif...	00:02	100%			
View more										

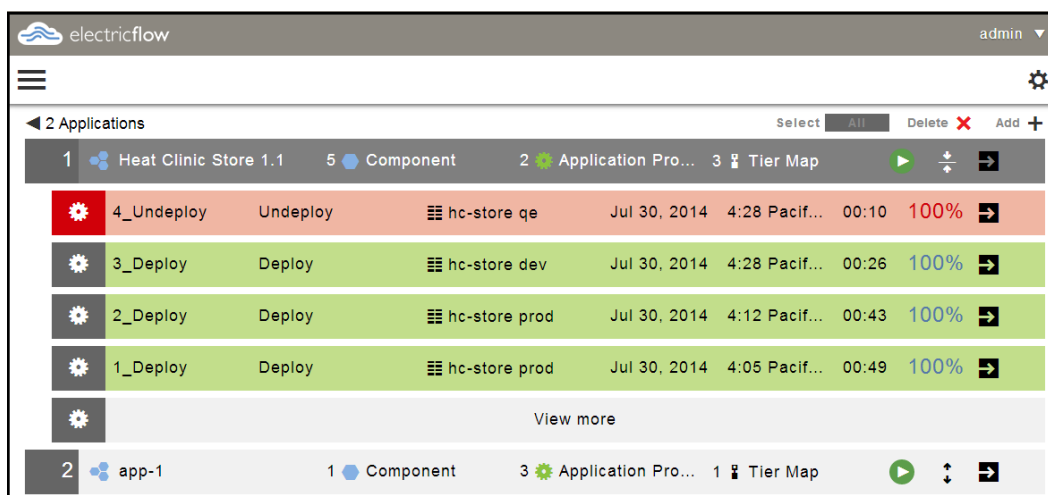
Click the right arrow in the 6_Test2 row to see the detailed results. The process has three steps, which were all successfully run.



The screenshot shows the ElectricFlow dashboard with a header bar containing the logo, the name 'electricflow', and a user dropdown menu set to 'admin'. Below the header is a navigation menu with a hamburger icon and a settings gear icon. The main content area is titled 'Applications' and includes a toolbar with 'Errors' and '0' buttons. The table lists the detailed test results for '6_Test2_app-1_Default_20140730...'. The row for '6_Test2_app-1_Default_20140730...' is highlighted in green, indicating a successful run. The row for 'S1' is highlighted in green, indicating a successful run. The row for 'S2' is highlighted in green, indicating a successful run. The row for 'S4' is highlighted in green, indicating a successful run.

Application	Component	Application Pr...	Tier Map	Test Name	Environment	Date	Time	Status	Progress	Action
	6_Test2_app-1_Default_20140730...	→ Success	Jul 30, 2014	4:18 Pacif...	00:01	100%				
	S1	→ Success	Jul 30, 2014	4:18 Pacif...	00:00	100%				
	S2	→ Success	Jul 30, 2014	4:18 Pacif...	00:00	100%				
	S4	→ Success	Jul 30, 2014	4:18 Pacif...	00:00	100%				

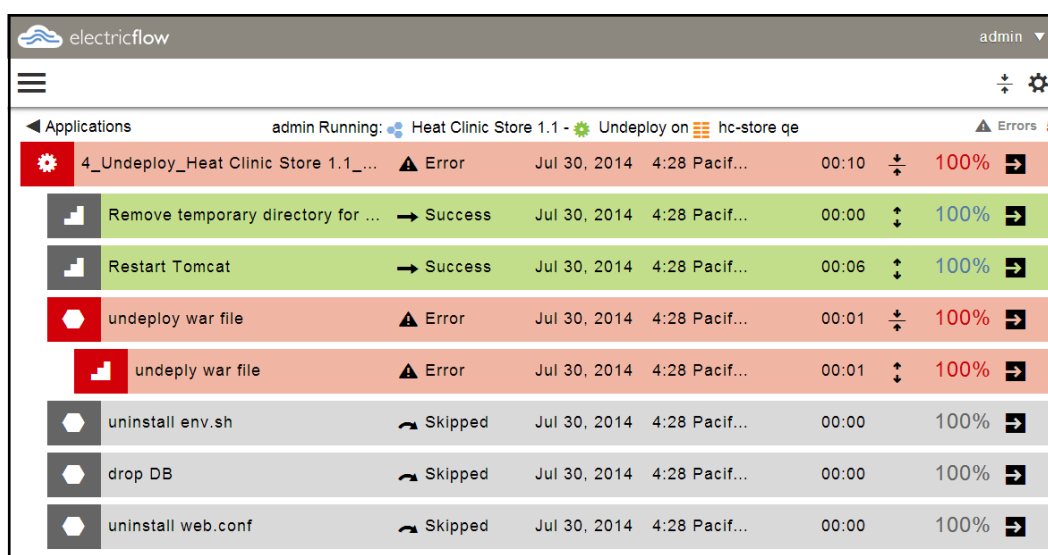
The following shows that the application called "Heat Store Clinic 1.1" did not run successfully on July 30, 2014, at 4:28 p.m.



The screenshot shows the ElectricFlow dashboard with a header bar containing the logo, the name 'electricflow', and a user dropdown 'admin'. Below the header is a navigation menu with a hamburger icon and a settings gear. The main content area is titled '2 Applications' and includes a toolbar with 'Select All', 'Delete', and 'Add' buttons. The applications are listed in a table with columns for application name, component, application profile, tier map, and deployment status. The first application is 'Heat Clinic Store 1.1' with 5 components. The second application is 'app-1' with 1 component. The deployment status for the first application is shown as a list of tasks: '4_Undeploy' (Undeploy), '3_Deploy' (Deploy), '2_Deploy' (Deploy), and '1_Deploy' (Deploy). Each task row includes a gear icon, a status icon (red for error, green for success), a task name, an action, a component, a date, a time, a duration, a progress bar, and a right arrow icon.

Application	Component	Application Profile	Tier Map	Task Name	Action	Component	Date	Time	Duration	Progress	Status
Heat Clinic Store 1.1	5	Application Pro...	Tier Map	4_Undeploy	Undeploy	hc-store qe	Jul 30, 2014	4:28 Pacif...	00:10	100%	Error
				3_Deploy	Deploy	hc-store dev	Jul 30, 2014	4:28 Pacif...	00:26	100%	Success
				2_Deploy	Deploy	hc-store prod	Jul 30, 2014	4:12 Pacif...	00:43	100%	Success
				1_Deploy	Deploy	hc-store prod	Jul 30, 2014	4:05 Pacif...	00:49	100%	Success
app-1	1	Component	3	Application Pro...	1	Tier Map					

Click the right arrow in the 4_Undeploy row to see the detailed results. The process has two steps that ran successfully, a component called "undeploy war file" with errors, three skipped objects, and five errors.



The screenshot shows the detailed results of the '4_Undeploy' task. The header bar is the same as the previous screenshot. The main content area is titled 'Applications' and includes a toolbar with 'admin Running: Heat Clinic Store 1.1 - Undeploy on hc-store qe' and an 'Errors 5' button. The tasks are listed in a table with columns for task name, status, date, time, duration, progress, and a right arrow icon. The tasks are: '4_Undeploy_Heat Clinic Store 1.1_...' (Error), 'Remove temporary directory for ...' (Success), 'Restart Tomcat' (Success), 'undeploy war file' (Error), 'undeply war file' (Error), 'uninstall env.sh' (Skipped), 'drop DB' (Skipped), and 'uninstall web.conf' (Skipped).

Task Name	Status	Date	Time	Duration	Progress	Status
4_Undeploy_Heat Clinic Store 1.1_...	Error	Jul 30, 2014	4:28 Pacif...	00:10	100%	Error
Remove temporary directory for ...	Success	Jul 30, 2014	4:28 Pacif...	00:00	100%	Success
Restart Tomcat	Success	Jul 30, 2014	4:28 Pacif...	00:06	100%	Success
undeploy war file	Error	Jul 30, 2014	4:28 Pacif...	00:01	100%	Error
undeply war file	Error	Jul 30, 2014	4:28 Pacif...	00:01	100%	Error
uninstall env.sh	Skipped	Jul 30, 2014	4:28 Pacif...	00:00	100%	Skipped
drop DB	Skipped	Jul 30, 2014	4:28 Pacif...	00:00	100%	Skipped
uninstall web.conf	Skipped	Jul 30, 2014	4:28 Pacif...	00:00	100%	Skipped

To troubleshoot the errors, you can click the right arrow in one of the "undeploy war file" rows and you go a Job Step Detail page in the automation platform.

Environment Inventory

How to get here: From the Environments List, choose an environment and click the **Inventory** button. The Environment Inventory for that environment appears.

The Environment Inventory is the state of the environment at a point in time.

- When an application is running, you can see the progress as it runs.
- After an application runs, you can see the details for the objects in application that ran in the environment.

Click the **View** details button in a row to see more details about a specific object including

- Environment name
- Application mapped to this environment
- Number of deployed artifacts in the application
- When the artifacts were deployed
- Status of the deployment: success or failure

In this example, the Environments List shows that an application was run on the "hc-store dev" environment and has one error.

4 Environments		Select	All	Delete	Add
1	Myenv	Enabled	0 Applications installed		
2	hc-store dev	Enabled	1 Applications installed	1	
3	hc-store prod	Enabled	1 Applications installed		
4	hc-store qe	Enabled	0 Applications installed		

The first level of the Environment Inventory appears.

Environments / Inventory		hc-store dev
1	Heat Clinic Store 1.1	6 Artifacts

In the second level, you can view more details. Click the View arrows at the end of the row.

The applications mapped to this environment appear.

Environments / Inventory						
1 Heat Clinic Store 1.1 6 Artifacts 1						
hc-warfile	com.myc...warfile	1.0	Tomcat	5 minute...		
env.sh	com.myc...nic:env	1.0	Tomcat	2 minute...	1/2	
database.conf	com.myc...:config		MySQL			
database.conf	com.myc...:config	1.0	MySQL	5 minute...		
database.sql			MySQL			
web.conf	com.myc...:config	1.0	Apache...	5 minute...		

To show more details in the third level, click the **Process** button at the end of the env.sh row.

Applications						
admin Running: Heat Clinic Store 1.1 - Deploy on hc-store dev Errors 11						
8_Deploy_Heat Clinic Store 1.1_Default_20140730...	Aborted	Jul 30, 2014 5:19 Pacifi...	00:54	100%		
Install Tomcat env settings	Error	Jul 30, 2014 5:19 Pacifi...	00:53	100%		
get files	Error	Jul 30, 2014 5:19 Pacifi...	00:52	100%		
place file	Error	Jul 30, 2014 5:19 Pacifi...	00:00	100%		
restart Tomcat	Error	Jul 30, 2014 5:19 Pacifi...	00:00	100%		
update web config	Skipped	Jul 30, 2014 5:19 Pacifi...	00:00	100%		
deploy app	Skipped	Jul 30, 2014 5:19 Pacifi...	00:00	100%		
update DB config	Skipped	Jul 30, 2014 5:19 Pacifi...	00:00	100%		
setup database	Skipped	Jul 30, 2014 5:19 Pacifi...	00:00	100%		
start tomcat	Skipped	Jul 30, 2014 5:19 Pacifi...	00:00	100%		
Create Link to application	Skipped	Jul 30, 2014 5:19 Pacifi...	00:00	100%		

In the fourth level, you can get more information for the steps in the Install component process by clicking on the "View details" arrow in the "get files," "place file," and "restart Tomcat" rows.

The screenshot shows the ElectricFlow interface with the 'Applications' tab selected. The header indicates 'admin Running: Heat Clinic Store 1.1 - Deploy on hc-store dev' and shows 'Errors 11'. The table lists various deployment steps and their outcomes.

Icon	Component Name	Status	Date	Time	Location	Duration	Progress	Errors
⚙️	8_Deploy_Heat Clinic Store 1.1_Default_20140730...	Aborted	Jul 30, 2014	5:19	Pacifi...	00:54	100%	11
🔧	install Tomcat env settings	Error	Jul 30, 2014	5:19	Pacifi...	00:53	100%	
📁	get files	Error	Jul 30, 2014	5:19	Pacifi...	00:52	100%	
📋	l1	Aborted	Jul 30, 2014	5:19	Pacifi...	00:00	100%	
📋	resource-01	Success	Jul 30, 2014	5:19	Pacifi...	00:01	100%	
📁	place file	Error	Jul 30, 2014	5:19	Pacifi...	00:00	100%	
📋	l1	Error	Jul 30, 2014	5:20	Pacifi...	00:00	100%	
📋	resource-01	Aborted	Jul 30, 2014	5:20	Pacifi...	00:00	100%	
🔄	restart Tomcat	Error	Jul 30, 2014	5:19	Pacifi...	00:00	100%	
📋	l1	Error	Jul 30, 2014	5:20	Pacifi...	00:00	100%	
📋	resource-01	Aborted	Jul 30, 2014	5:20	Pacifi...	00:00	100%	

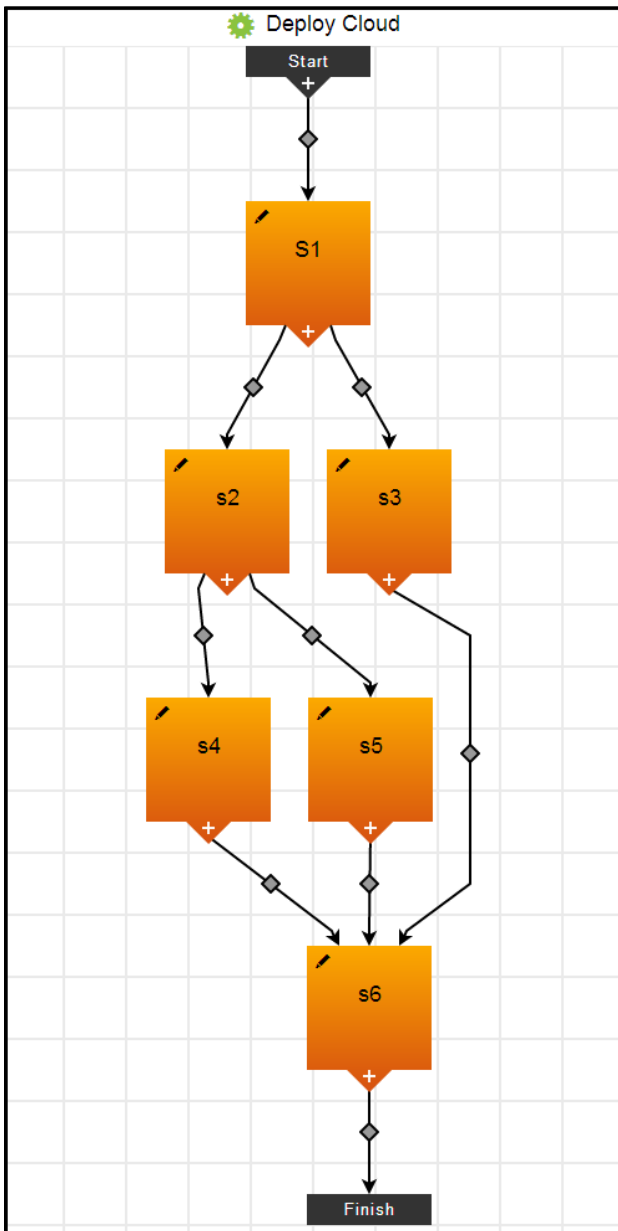
The Environment Inventory shows this information:

- Environment name: hc-store dev
- Name of the application mapped to this environment: Heat Clinic Store 1.1
- Components in the application: See the second level.
- Number of the artifacts associated with components: Six. For details, see the first and second levels.
- Each component has an artifact with a version number: See the second level.
- Each component is also in an application tier: See the second level.
- Time when the artifact was deployed: See the third and fourth levels.
- Error counts if there are any errors: See the third and fourth levels.
- Number of resources that are successfully assigned to applications on a per-artifact basis: See the third and fourth levels.

Viewing Job Details

When you run a process in an application, the system records the results of the process as each step is performed. You can see a summary of the results in the Applications Inventory.

This example shows the results of running the Deploy Web application process, which consists of the Deploy Cloud component process.



When you run the Deploy Web process, you can see the status of the process as it runs in the Application Inventory.

You can also see which application processes were run, with the latest process appearing first.

5 Applications

Select

All

Delete

Add

1

Corvet 3.0

1 Component

0 Application Process

0 Tier Map

2

Crystal's App

1 Component

0 Application Process

0 Tier Map

3

Heat Clinic Store 1.1

5 Component

2 Application Process

3 Tier Map

4

Web Servers

1 Component

1 Application Process

1 Tier Map

2_Deploy Web

Deploy Web

hc-store dev

Jul 28, 2014 5:55 Pacif...

00:04

100%

1_Deploy Web

Deploy Web

hc-store dev

Jul 28, 2014 5:54 Pacif...

00:04

100%

5


app-1

1 Component

1 Application Process

1 Tier Map

To see more details about the process, click the process about which you want more information. If you want to learn more about the 2_Deploy Web process, click **2_Deploy Web** in the Applications Inventory. It now shows the details for the process that you selected.

electricflow

Notice the order of the process steps in the Applications Inventory. The order is not based on the branches in the process. Instead, the system lists the steps based on their level in the process.

- Step S1 is the first step in the process and is listed first. There are no other steps at this level so the next step is in the level following S1.
- The next step in the list is s2.
- Step s3 is on the same level as s2. There are no other steps at this level so the next step is in the level after this.
- The next step is s4.
- Step s5 is on the same level as s4. There are no other steps at this level so the next step is in the level after this.
- The last step is s6.

Change Tracking

Change Tracking is designed to track every change between every state of non-runtime ElectricFlow objects and to allow you to export or revert to any previous state of these objects. ElectricFlow tracks the changes to tracked objects including applications, procedures, workflows, workspaces, resources, and project-owned components such as library components and records a *Change History* of the historical states of the system and the changes between them. The change history has a copy of every state in which every non-runtime object has been, the object's current state, and indexing records for searching through the database. The tracked objects are not affected when ElectricFlow executes an object that is usually created or modified at execution time, such as a workflow or process.

Change tracking allows you to do the following:

- When debugging a failed job or looking for more information about a component in an ElectricFlow, see the Change History for the changes relevant to that object.
- When searching for specific change history records, filter the records by time frame, change type, entity type, or developer.
- Revert changes to an object or to an objects and its children.
- When you want to determine the differences between objects, export them at various levels in the object hierarchy.

In ElectricFlow, you can use Change Tracking in all aspects of the ElectricFlow end-to-end solution:

- In build-test automation, track the Change History of artifacts, resources, and ACLs.
- In deployment automation, track the Change History of components, environment tiers, and process steps (application or component). You also use Change Tracking with snapshots to make it easier to deploy reliable and repeatable software for Continuous Delivery.
- In pipeline automation, track the Change History of stages and tasks.

More about application, deploy, and run:

As you use ElectricFlow, remember that these terms have different meanings within ElectricFlow *and* outside of ElectricFlow when you deploy your software or application:

Term	Within ElectricFlow	Outside of ElectricFlow
Application	The application that you design and run (deploy) to produce your software for continuous delivery across different pipelines.	The software, system or application that you build, test, install, implement, release, and deploy using ElectricFlow. This is the end product of using ElectricFlow.
Deploy	Running the application that you designed in ElectricFlow. The end product is your software, system, or application. Deploy is a synonym of run in ElectricFlow.	All the processes or actions to develop and run your software in its environment, including building, testing, implementing, installing, configuring, making changes, and releasing.
Run	Running the application that you designed. The end product is your software, system, or application. <i>Run</i> is a synonym of <i>deploy</i> in ElectricFlow.	All the processes or actions to use software in its environment, including implementing, installing, configuring, debugging, troubleshooting, and releasing.

Performance Consequences of Change Tracking

Change Tracking affects how ElectricFlow runs. The impact can be from low (minimal changes to the times to update the database) to high (significantly longer times to update the database).

ElectricFlow needs to store additional information in the database about the tracked objects in the Change History. In addition to storing normal information about ElectricFlow operations, ElectricFlow stores a separate archival copy of every state of every tracked object as well as the indexing information used to retrieve the Change History of tracked objects and the objects they own. This increase in database usage has performance consequences.

For most of ElectricFlow operations, the performance consequences of Change Tracking are insignificant, typically a few percent slower in one of these situations:

- None of the affected objects are tracked (job records and similar run time-only objects are not tracked).
- Only a small number of tracked objects are changed by operations.

However, for operations that affect a very large number of tracked objects in a single operation (such as importing, cloning, or deleting a large project for which change tracking is enabled), the performance consequences *will be* significant. The performance consequences are similar when enabling or disabling Change Tracking for a large project, because both operations require the server to write to the database the records for every tracked object owned by the project.

For these Change Tracking-intensive operations, the system performance will typically be significantly slower because several times as much data is being read and written between the ElectricFlow server and the

database. In our testing, we have seen a wide range of degrees of performance decrease from 1.25x to 10x slower when Change Tracking is enabled on a large project, compared to a project on which Change Tracking is disabled. The decrease in performance is typically from 2x to 6x.

Use Case 1: Importing and Cloning Large Projects with Change Tracking Enabled

ElectricFlow does the following to increase the performance of Change Tracking-intensive operations in a large project with this configuration:

- The project contains over 20,000 audited objects.
- Change Tracking is enabled.

To increase the performance when a large project is imported or cloned, ElectricFlow automatically decreases the amount of Change History indexing information that it saves, reducing the level of detail for these operations in the Change History. While this can improve the system performance, it can also make it harder when you want to revert an object to a specific state or to find information in the Change History while troubleshooting or debugging an issue. If you want suppress this behavior, use

`--reducedDetailChangeHistory false` in the `import` or `clone` API command.

By default, the objects listed in the table below and the tracked objects that they own are always tracked when Change Tracking is enabled (except when Change Tracking is globally disabled).

During an upgrade from ElectricFlow 5.0.x to ElectricFlow 6.0, when the ElectricFlow server starts, the server writes to the database a record for every tracked object and the objects that it owns. The objects listed in the table get new audit records. If any of the objects owns many tracked objects (such as properties), the startup process will be slow.

Objects (Entities)	Path
server	/server
system objects	/systemObjects/*
users	/users/*
groups	/group/*
plugin	/plugins/*
email configuration	/emailConfigs/*
workspaces	/workspaces/*
zones	/zones/*
gateways	/gateways/*

Objects (Entities)	Path
agents	/agents/*
resources	/resources/*
resource pools	/resourcesPools/*
repositories	/repositories/*
directories	/directories/*

Use Case 2: Change Tracking of Non-Project-Owned Objects

In this context, *non-project-owned objects* include all objects not contained within a project. Such objects include resources, artifacts, users, groups, zones, and other objects as well as the property sheets, properties, access control lists (ACLs), and ACL entries (ACEs) for them. In releases earlier than ElectricFlow 6.2, Change Tracking was enabled by default for non-project-owned objects, and there was no way to disable Change Tracking for them. When users upgraded from earlier versions of ElectricFlow to ElectricFlow 6.0.x, the upgrade process initialized and created Change Tracking-related data for them in the database. The performance impact was usually low, except when there was a very large number (in the several hundred thousands) of non-project-owned objects, such as many properties under the `/server` directory; in this situation, the performance of the first startup was significantly slower. The best solution to this situation is to relocate most of the projects under a project and disable Change Tracking for that project, which may not be a practical short-term solution.

Starting in ElectricFlow 6.2, Change Tracking can be disabled for all non-project-owned objects. (Change Tracking for these objects is enabled by default.) Using this feature is not recommended as a long-term solution; instead we recommend moving most of the objects under a project that has Change Tracking disabled. (Repeatedly disabling and enabling Change Tracking for non-project-owned project is expensive and time-consuming.)

In ElectricFlow 6.2, when there are many non-project-owned objects, the best way to disable Change Tracking for all non-project-owned objects is to change the `database.propertiesconfiguration` file to disable Change Tracking for these objects before starting the server running a pre-ElectricFlow 6.2 release. In the `database.propertiesfile`, make sure that it has this line:

```
COMMANDER_DB_NON_PROJECT_AUDITING_ENABLED=false
```

Then install ElectricFlow 6.2 and start the server, running the system this way until most of the non-project-owned objects are relocated to a non-tracked project, and re-enable Change Tracking for non-project-owned objects, which requires restarting the server.

Estimating Database Growth

Here is one way to estimate the rate at which the database grows for Change Tracking:

Most of the information needed to estimate the database growth rate can also be parsed from an exported XML file of the project (such as a full import), except for the version count.

The likely rate of database growth is based on these factors:

- How many objects of different types will be tracked
- How big they are
- How often their state changes

The rate of database consumption is then the sum over all different types of tracked objects of the product of these three factors.

Common Usage Patterns

For most customer usage patterns, the most common objects (the ones that dominate on first factor) are ACLs and their ACEs, properties, or all of these objects. Typically:

- ACLs and ACEs: ACLs are consistently small, and usually have only a few ACEs.
- Properties: While most of them are small, some properties can be very large. The average property size is large.

For most users, a good quick estimate can be based on properties, especially properties that are large or are modified often.

Estimating Process

If you expect to enable Change Tracking for all projects, determine the values for these factors and calculate the database growth rate:

Database growth rate = Factor 1 x Factor 2 x Factor 3

- Factor 1: How many objects of different types will be tracked

The following procedure is only for properties:

1. Join the `ec_property` table in the database joined to the `ec_property_sheet` table by `ec_property.parent_sheet_id = ec_property_sheet.id`.
2. Search for properties where the `ec_property_sheet.entity_type` value is one of the entity types corresponding to a database table that has a matching `ec_*_aud` table.
3. Determine the number of tracked objects per property type.

For most usage patterns, the great majority of properties will belong to entity types like `job` and `jobStep` that are not change tracked and so have no corresponding `ec_*_aud` table.

If you are already using ElectricFlow 5.3 or later, these properties and properties belonging to projects for which Change Tracking is disabled are also recognizable by having `ec_property_tracked` set to `false`.

- Factor 2: How big they are (size)

The following procedure is only for properties.

For small properties:

1. Search for objects that are less than 450 bytes in the `ec_property.string` column.
2. Add an overhead value of half a kilobyte per property for the other columns.
3. Determine the average size of the tracked objects.

For larger properties:

1. Search for the length of the value in the `ec_clob` table linked to by the `ec_property.clob` column.
2. Add an overhead value of one kilobyte per property for the other columns.
3. Determine the average size of the tracked objects.

- Factor 3: How often new copies of the objects are generated

The following procedure is only for properties:

1. Use one of these methods:
 - Determine ratio of how long ago the `ec_property.created` and `ec_property.modified` datetimes are , which gives an estimate concentrating on only the most recent changes.

and/or
 - Determine the value of the `ec_property.version` counter divided how long ago the `ec_property.created` datetime is, which gives a long-term average rate of changes to the property since it was created.
2. Determine the average update frequency.

For properties that are frequently updated always to be a numerical value, it is possible to suppress Change Tracking of numerical-value updates by using one of these commands:

```
ectool: ectool modifyProperty <projectName> -- path <propertyPath> -- counter true
```

or

```
ec-perl: $cmdr-> modifyProperty (<projectName>, {path => <propertyPath>, counter => true});
```

When the `counter` flag is set for a property, Change Tracking does not track changes to the property if the only change was a change in the property value from one numeric value to another. All other forms of changes to the properties that have this flag set are tracked normally.

CAUTION: Reverting an object that owns a counter property to a previous state will revert the value of the counter property to its value at the previous time that a change to this property was tracked—typically this is when its value is initialized. This may not be the desired behavior, so you may need to manually set a counter property if it is reverted.,.

Alternative Estimating Methods

Depending on your usage patterns, one of these methods may result in a more accurate estimate than the previous method of determining the average factor values and multiplying them

Split the set of properties of change tracked objects into different populations

For each object, perform the previous method of determining the average factor values and multiplying them per tracked object.

Add the values for each tracked objects together.

Example:

- If you have one set of properties (such as those in one project, or belonging to one user) containing very large properties that are never changed, and another set containing very small properties that change often, multiplying the average property size by the average rate of property update would result in an overestimate of the total database consumption.
- If you put the properties in separate projects and calculate the database growth rate for each project, the database growth rates are more accurate.

Other columns that may be of interest include `ec_property.owner` (who created the property) and `ec_property.last_modified_by` usernames (who last modified the property).

Best Practices for Change Tracking

Change Tracking allows you to troubleshoot and debug issues, find information about specific objects, and revert an object to a specific state. However, Change Tracking affects how the system performance. In some situations, the performance consequences of Change Tracking can be significant.

This section describes how to improve the system performance.

When to Enable or Disable Change Tracking for a Project

In some situations, enabling or disabling Change Tracking on a project causes significant performance consequences. The ElectricFlow system performance becomes slower as the server writes to the database the records for all the tracked objects in the project.

To determine when to use Change Tracking on a project, do the following:

1. Export the project using this command:

```
ectool export <project_name> --excludeJobs 1 --relocateable 0 --withAcls 1 - withNotifiers 1
```

2. Evaluate the results.

a. The size of the results file gives you a good estimate of how big the project is:

- If the file size is in the hundreds of MB, the project is large.
- If the file size is in the tens of MB or less, the project is small.

b. (Optional) Write a script to parse the results and determine these statistics:

- Total number of objects that can be tracked in the project.

Use this value to determine the project size (for example, small, medium, or large), which is used to determine how long it would take to enable, disable, copy, import, or delete a project on which Change Tracking is enabled.

- The total size (in XML format) of the entities in the project that can be tracked by Change Tracking.

This gives a rough estimate of how much the database usage will immediately grow when Change Tracking is enabled for the project.

- The rate at which tracked entities are modified in the project.

Look at the most recent last-modification dates of entities owned by the project for a pattern where certain entities are modified on a regular basis.

If there are objects that appear to be being modified frequently, the script can also parse out their path and the user name they were last modified by.

Next Steps:

- If the project is small, and its contents are not frequently changed by an automated process, you can safely enable Change Tracking for it.
- If the project is large, and if you frequently export, import, or copy it, and if the performance of these operations is very important to you, enabling Change Tracking for it may be problematic (see [Estimating Process](#) on page 427 for a possible workaround).
- If significant parts of the contents of the project are frequently updated by an automated process, enabling Change Tracking for the project will consume significantly more database space, and tracking the changes performed by the automated process is unlikely to be of much value, so enabling Change Tracking for the project is inadvisable.

Splitting Tracked Objects Into Separate Projects

If you export the project, evaluate the results, and determine that enabling or disabling Change Tracking on the project will significantly affect the performance of the ElectricFlow system, one way to improve the system performance is to split the project into separate projects based how the objects are modified.

For example, if you parse the results and find that there are two types of entities in your current project:

- Objects that are manually modified over time by real users.

These objects represent real changes in project and are modified as procedures, jobs, and applications are run.

- Objects that are frequently updated at regular intervals by an automated process using metaprogramming.

These objects are automatically generated and updated on a regular basis. The project file should have a large list of modifications to tracked objects.

Then you should split the objects into separate projects.

- Put the objects that are manually modified by real users in one project and enable Change Tracking on this project.
- Put the objects that are updated by an automated process in a different project and disable Change Tracking on it.

Configuring Change Tracking

Change Tracking must be enabled when ElectricFlow starts for your system to track changes and record the Change History.

By default, Change Tracking is enabled for projects created in ElectricFlow 5.3 or later and for projects created before upgrading to ElectricFlow 5.3. When you import a project created before ElectricFlow 5.3, Change Tracking is disabled for the project.

Enabling Change Tracking Globally

When installing ElectricFlow:

1. Add this line to the `database.properties` file:

```
COMMANDER_DB_AUDITING_ENABLED=true
```

2. Restart the ElectricFlow server.

Enabling Change Tracking on a Per-Project Basis

IMPORTANT: Unless the `allowEnablingDisablingChangeTrackingProjectLevel` setting has been altered, only admin users are allowed to perform this task. The default setting is `adminOnly`. If the setting is changed to `anyoneWithAccess`, any user with access to the project can enable or disable Change Tracking for the project

You can enable Change Tracking one of the following ways:

In the ElectricFlow Platform UI

Change Tracking is enabled when the **Enable Change Tracking** check box is selected.

New Project

Name:

Enable Change Tracking: ☒

Description:

Default Resource: Browse

Default Workspace: Browse

To disable Change Tracking, click the **Enable Change Tracking** check box to clear it, and click **OK**.

Using ectool

- Enter `ectool modifyProject <projectName> --tracked true` to enable Change Tracking.
- Enter `ectool modifyProject <projectName> --tracked false` to disable Change Tracking.

Using ec-perl:

- Enter `$cmdr->modifyProject(<projectName>, {tracked => true});` to enable Change Tracking.
- Enter `$cmdr->modifyProject(<projectName>, {tracked => false});` to disable Change Tracking.

For properties that are frequently updated always to be a numerical value, it is possible to suppress Change Tracking of numerical-value updates by using one of these commands:

```
ectool:ectool modifyProperty <projectName> -- path <propertyPath> -- counter true
```

or

```
ec-perl: $cmdr-> modifyProperty (<projectName>, {path => <propertyPath>, counter => true});
```

When the `counter` flag is set for a property, Change Tracking does not track changes to the property if the only change was a change in the property value from one numeric value to another. All other forms of changes to the properties that have this flag set are tracked normally.

CAUTION: Reverting an object that owns a counter property to a previous state will revert the value of the counter property to its value at the previous time that a change to this property was tracked—typically this is

when its value is initialized. This may not be the desired behavior, so you may need to manually set a counter property if it is reverted,.

Upgrading to ElectricFlow 6.x

Change Tracking is enabled when you upgrade to ElectricFlow 6.x. This can significantly increase the time it takes to complete the upgrade.

If you want to upgrade with Change Tracking disabled, add this line to the `database.properties` file before starting the upgrade:

```
COMMANDER_DB_AUDITING_ENABLED=false
```

Customizing the Change History Page

The performance of the Change Tracking feature is affected by number of records in the Change History as well as the number of entries being tracked. For example, a page showing 5000 entries may be slow to load and update and does not provide much useful information.

This ElectricFlow server uses the lowest of the following limits to determine the maximum amount of records to display in the Change History page:

- Maximum amount of records on the Change History page

To change the maximum number of records in the Change History page:

1. Set the `CHANGE_TRACKING_HARD_MAX_RECORDS` parameter in the `wrapper.conf` file to a new value.

The default value is 1000.

2. Restart the ElectricFlow server.

- Maximum number of records retrieved

Set the `TrackingMax Records` server setting to a new value not exceeding the `CHANGE_TRACKING_HARD_MAX_RECORDS` parameter in the `wrapper.conf` file.

To set `TrackingMaxRecords`, do one of the following:

Change the value in the UI. See the Server Settings page in the automation platform UI.

Use `ectool` to change the value. For example, enter the following command to limit the number of records retrieved to 100:

```
ectool setProperty /server/settings/changeTrackingMaxRecords --value 100
```

Searching the Change History

Follow these steps to start a search in the Change History.

You can start a Change History search from most pages in the ElectricFlow UI.

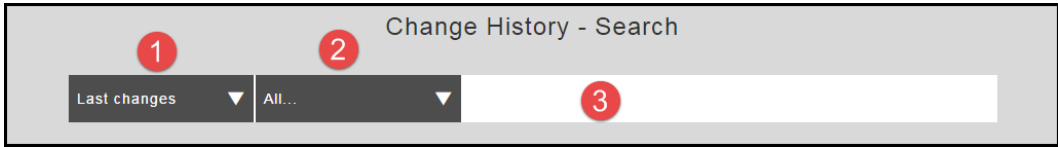
1. Click the **Search** button or click **Change History**.

Example:



The **Change History - Search** dialog box opens.

Example:

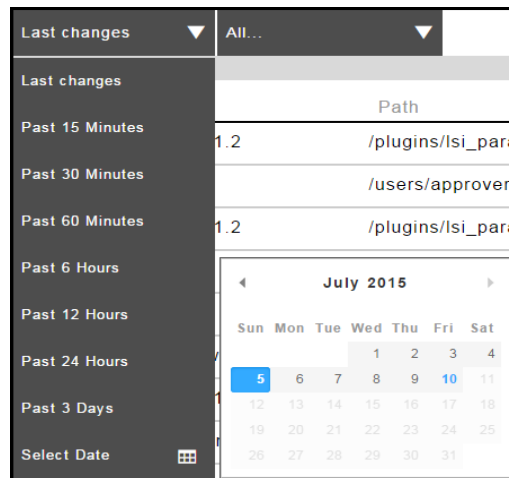


1	<p>Time range field.</p> <p>Click the down arrow to open the drop-down list of start times.</p> <p>The end time is the current time.</p>
2	<p>Objects field.</p> <p>Click the down arrow to open the drop-down list of objects to include in the search. You can select All or specific objects.</p> <p>By default, seven of the most commonly tracked objects are selected.</p>
3	<p>Search criteria.</p> <p>After you type, the system starts searching for objects based on the time range and objects that you selected.</p> <p>The search results are in the Change History.</p>

2. To select a time range for the search:
 - a. Click the down arrow in the **Time range** field to open the drop-down list.
 - b. Select a time range.
 - c. If you want to use a time increment longer than three days, do the following:
 - a. Click **Select Date**.

The Date Picker opens.

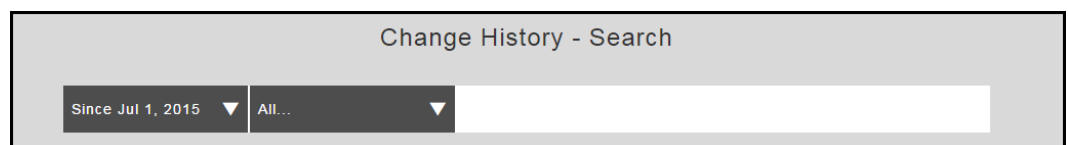
Example:



- b. Select a date.

The Date Picker closes and the date that you selected appears in the Time Increment field.

Example:



3. To select an objects for the search:
 - a. Click the down arrow in the **Object** field to open the drop-down list.
 - b. Select the objects for the search.
4. Enter the search criteria.

As you type, the system starts to search for objects that match your search criteria.

A list of objects matching your search criteria appears in the results section.

5. Select an object in the list.

Example:

Since Jul 1, 2015	All...	property	X
Access Control En...	ecscm-property-2.0.1.6...	/plugins/ecscm-property-2.0.1.65837	
	propertyviewer-1.0.1	/plugins/propertyviewer-1.0.1	
Access Control List	propertyviewer-1.0.1	/plugins/propertyviewer-1.0.1	
	ecscm-property-2.0.1.6...	/plugins/ecscm-property-2.0.1.65837	
Plugin	statetemplate_fullprope...	/server/ec_notifiertemplates/statetemplate_fullpropertypaths	
	propertyviewer-1.0.1	/plugins/propertyviewer-1.0.1	
Property	ecscm-property-2.0.1.6...	/plugins/ecscm-property-2.0.1.65837	
	artifactversionlocationp...	/projects/default/applications/heat clinic (killer app)/component...	
	artifactversionlocationp...	/projects/default/applications/jpetstore-aws-beanstalk/compone...	
	label	/server/ec_notifiertemplates/statetemplate_fullpropertypaths/la...	
	artifactversionlocationp...	/projects/default/applications/heat clinic store 1.1/components/...	
	artifactversionlocationp...	/projects/default/applications/heat clinic (killer app)/component...	
	artifactversionlocationp...	/projects/default/applications/heat clinic store 1.1/components/...	
	artifactversionlocationp...	/projects/default/applications/heat clinic store 1.1/components/...	
	series 1 property function	/projects/electric cloud/schedules/report recent job trend/series...	
	artifactversionlocationp...	/projects/default/applications/heat clinic (killer app)/component...	

The change history for the object that you selected appears.

Example:

Change History for propertyviewer-1.0.1						
<div> <div>▼ Last changes</div> <div> <div>7/10/15 - 1:02 PM</div> <div>Now - 7/10/15 - 6:18 PM</div> <div>Most Recent</div> </div> </div>						
View All Changes	When	What	Name	By...	Change	Path
Objects						
✓ Acl (1)	1 Jul 10, 2015	1:02 PM Pacific...	acl	propertyviewer-1.0.1	admin	created
✓ Ace (3)	2 Jul 10, 2015	1:02 PM Pacific...	ace	propertyviewer-1.0.1	admin	created
Changes						
✓ Created (4)	3 Jul 10, 2015	1:02 PM Pacific...	ace	propertyviewer-1.0.1	admin	created
Changed by...						
✓ Admin (4)	4 Jul 10, 2015	1:02 PM Pacific...	ace	propertyviewer-1.0.1	admin	created

Modifying the Change History

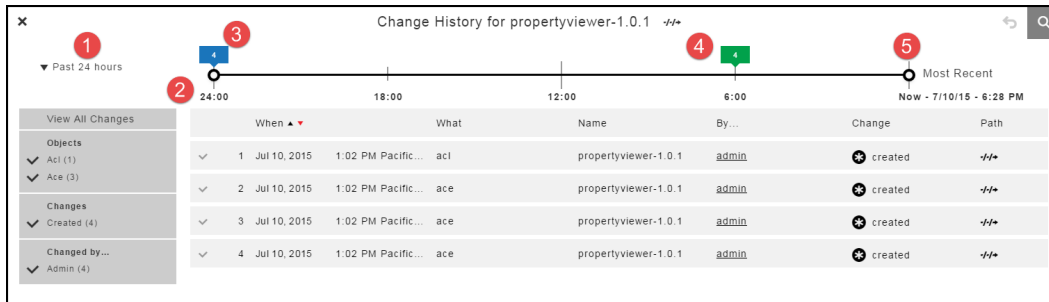
You can modify the information that appears in the Change History with these settings:

- Time increment—Go to [Change History Time Line](#) on page 956.
- Time line—Go to [Change History Time Line](#) on page 956.
- Filters—Go to [Change History Filters](#) on page 962.

Change History Time Line

The time line is at the top of the Change History page.

This example shows the Change History for a property called *propertyviewer-1.0.1*.



1	<p>Time increment.</p> <p>The default is Last Changes.</p> <p>Click the down arrow to select another time increment.</p>
2	<p>The system automatically calculates the minutes, hours, and days since the last successful run.</p> <p>In the example, the last successful run occurred 24 hours ago. The time line is divided into four 6 hour subdivisions.</p>
3	<p>Total number of changes in the selected time increment.</p>
4	<p>The number of changes that occurred between 6 hours ago and now is 4.</p> <p>When you click the change number, the Change History is updated and shows only those changes.</p>
5	<p>Drag the start and end time markers to view specific changes.</p>

Default Settings

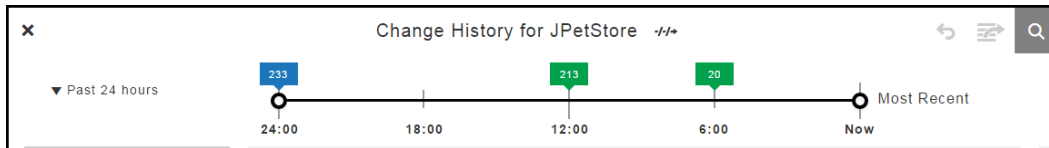
The default time increment is **Last Change**.

The entire time line is displayed, and all the changes are in the list below the time line.

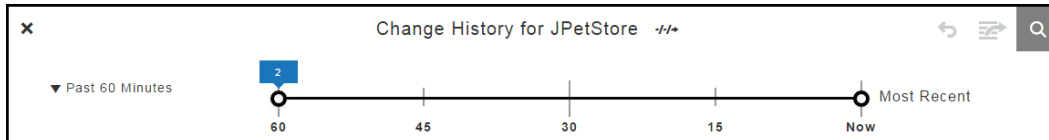
Number of Changes

The time line shows the number of changes throughout the time increment. In the following example:

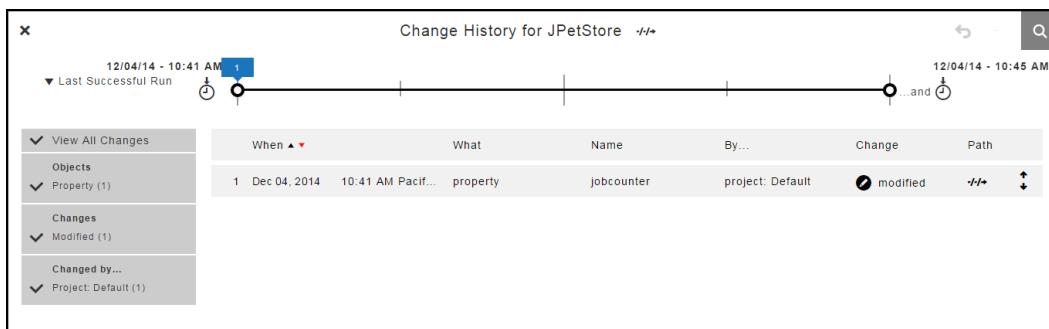
- There have been 233 changes in the last 24 hours.
- There have been 213 changes in the last 12 hours.
- There have been 20 changes in the last 6 hours.



When you change the time increment, there have been two changes in the previous 60 minutes.



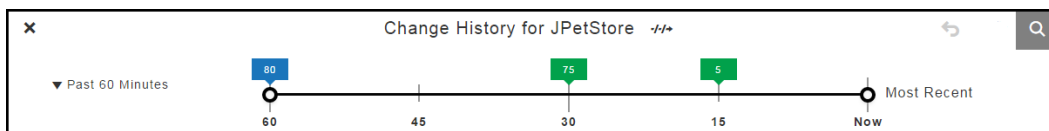
Time Increment



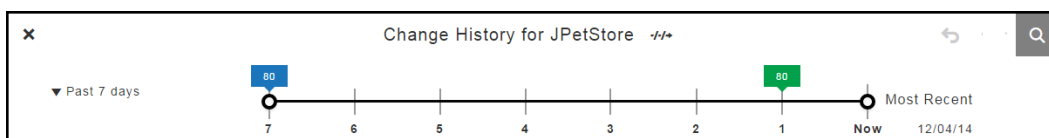
The system automatically determines how the time line is divided for the selected time increment.

When the range is changed to **Past 60 Minutes**, the time line changes:

- The start time is 60 minutes from **Now**.
- The end time is when the **Most Recent** change occurred (**Now**).
- The time line has four divisions.



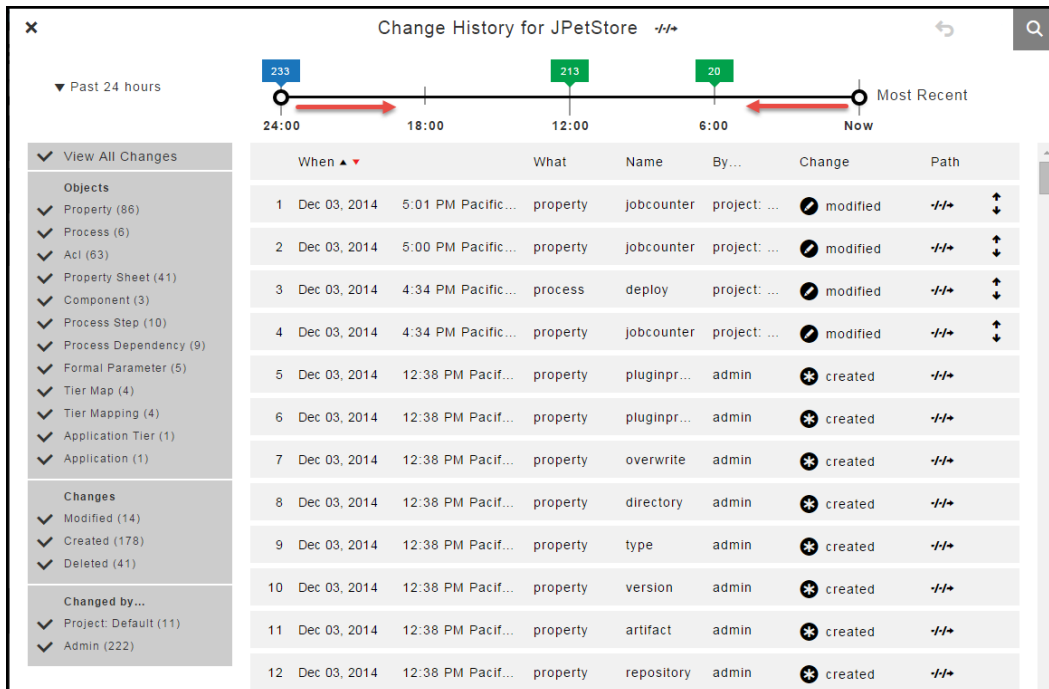
If the increment is **Past 7 Days**, the time line has seven one-day divisions.



Moving the Start and End Times Manually

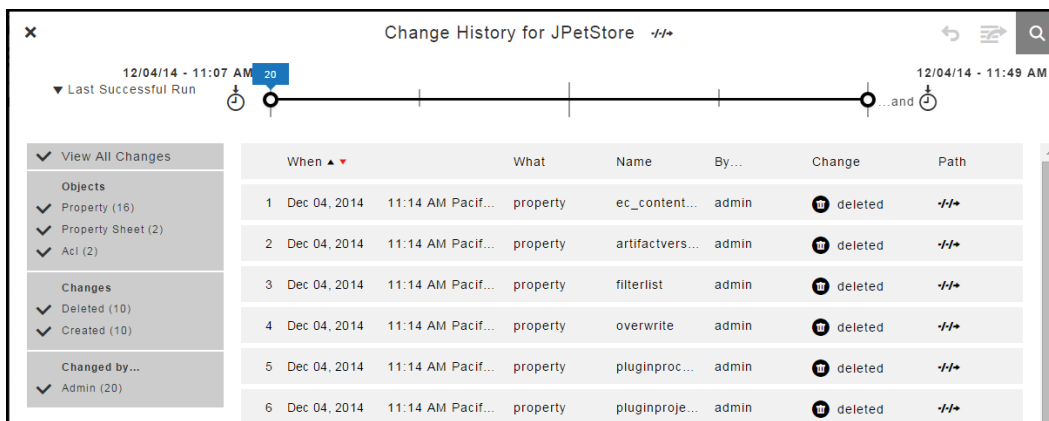
You can manually move the start and end times on the time line.

When you move the start time to 18:00 and the end time to 6:00, the list of objects in the change history changes.



Setting Custom Time Increments

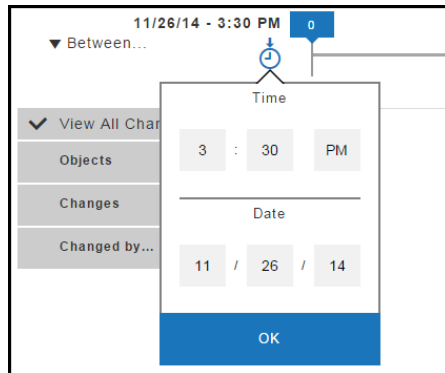
Example:



1. Select **Between**.

A drop down dialog box opens.

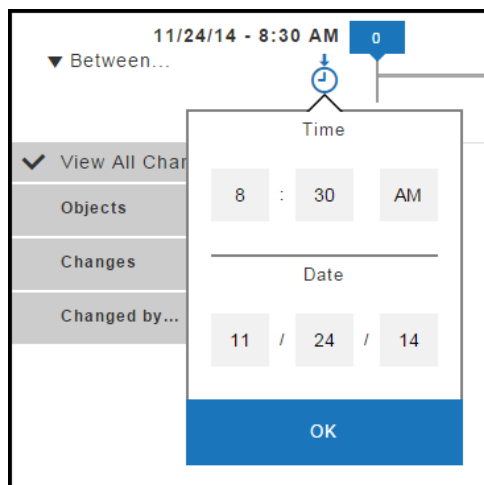
Example:



2. Select the time and date for the start of the time line.

The default settings are **3:30 PM** and eight days before the current date.

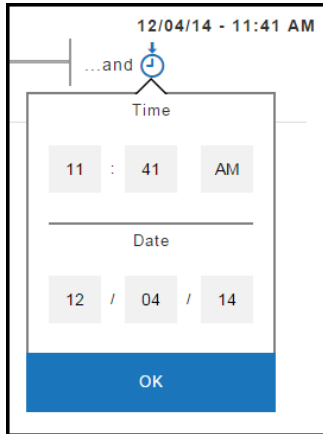
Example:



3. Click **OK**.

A drop down dialog box opens at the other end of the time line.

Example:

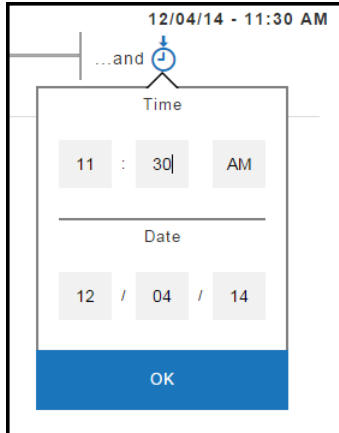


The screenshot shows a dialog box titled "Time" with a date and time selection interface. At the top, it displays "12/04/14 - 11:41 AM". Below this, there is a section for "Time" with input fields for "11", ":", "41", and "AM". Below the time section is a section for "Date" with input fields for "12", "/", "04", "/", and "14". At the bottom of the dialog is a blue button labeled "OK".

4. Select the time and date for the end of the time line.

The defaults are **3:30 PM** and the current date.

Example:



The screenshot shows the same dialog box as before, but with the time set to "30" and "AM". The date remains "12 / 04 / 14". The "OK" button is still at the bottom.

5. Click **OK**.

The time line changes to show only the changes from the start and end times and dates that you selected.

Example:

Change History for JPetStore

11/24/14 - 8:30 AM 100 100 12/04/14 - 11:30 AM

Between... and...

View All Changes	When ▲ ▼	What	Name	By...	Change	Path
Objects	1 Dec 04, 2014 10:28 AM Pacific...	acl	jpetstore	admin	* created	././
▼ Acl (25)	2 Dec 04, 2014 10:28 AM Pacific...	acl	jpetstore	admin	* created	././
▼ Property Sheet (16)	3 Dec 04, 2014 10:28 AM Pacific...	propertySh...	jpetstore	admin	* created	././
▼ Application (1)	4 Dec 04, 2014 10:28 AM Pacific...	propertySh...	ec_deploy	admin	* created	././
▼ Property (41)	5 Dec 04, 2014 10:28 AM Pacific...	application	jpetstore	admin	* created	././
▼ Application Tier (1)	6 Dec 04, 2014 10:28 AM Pacific...	property	ec_notifier...	admin	* created	././
▼ Component (1)	7 Dec 04, 2014 10:28 AM Pacific...	property	ec_deploy	admin	* created	././
▼ Process (2)	8 Dec 04, 2014 10:28 AM Pacific...	acl	ec_deploy	admin	* created	././
▼ Process Step (2)	9 Dec 04, 2014 10:28 AM Pacific...	acl	apptier	admin	* created	././
▼ Formal Parameter (3)	10 Dec 04, 2014 10:28 AM Pacific...	propertySh...	apptier	admin	* created	././
▼ Tier Map (4)	11 Dec 04, 2014 10:28 AM Pacific...	acl	apptier	admin	* created	././
▼ Tier Mapping (4)						
Changes						
▼ Created (86)						
▼ Modified (4)						
▼ Deleted (10)						
Changed by...						
▼ Admin (95)						
▼ Project: Default (5)						

Change History Filters

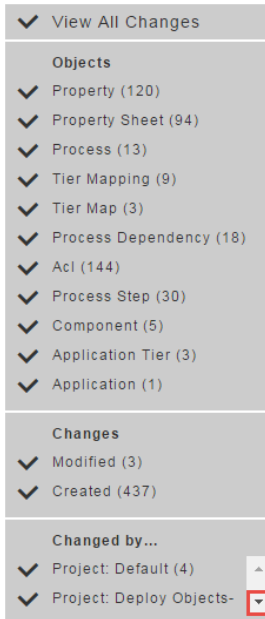
You can use filters to view changes to specific objects, the types of changes, and the users how made those changes.

Instead of selecting **View All Changes**, you can select specific objects, such as only properties, processes, property sheets, process steps, and process dependencies that have been modified by the Project:Default and Admin users.

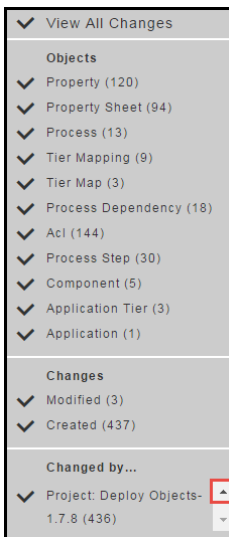
View All Changes	When ▲ ▼	What	Name	By...	Change	Path
Objects	1 Dec 03, 2014 4:34 PM Pacific...	property	jobcounter	project: ...	modified	././
▼ Property (43)	2 Dec 03, 2014 4:34 PM Pacific...	process	deploy	project: ...	modified	././
▼ Process (2)	3 Dec 03, 2014 11:35 AM Pacific...	process	deploy	admin	modified	././
▼ Property Sheet (20)						
▼ Acl (27)						
▼ Component (1)						
▼ Process Step (6)						
▼ Process Dependency (8)						
Changes						
▼ Modified (6)						
▼ Created (60)						
▼ Deleted (41)						
Changed by...						
▼ Project: Default (2)						
▼ Admin (105)						

When the list of filter criteria is long, not all of the criteria may appear in the filter list. To see all of the criteria, use the up or down arrows to see all the options.

This list does not show all of the users. Use the up and down arrows to see all four of the users.



Click the down arrow to see the other users.



Viewing the Change History

You can open and view the change history from the following objects in ElectricFlow:

- Applications List
- Applications Visual Editor
- Application Process Designer
- Artifacts
- Component
- Component Process Designer
- Environments Designer
- Environment Tier
- Jobs
- Process Step (Application or Component)
- Projects
- Resources
- Workflows

Viewing the Change History from the Applications List

You may want to view the Change History for these objects in an application:

- An application process that did not run successfully

4	JPetStore	1 Component	1 Application Process	4 Tier Map				
5	Test	1 Component	1 Application Process	1 Tier Map				
7_Proc	Proc	Env	Dec 04, 2014 11:08 Pac...	00:06	100%			
6_Proc	Proc	Env	Dec 04, 2014 10:45 Pac...	00:07	100%			
5_Proc	Proc	Env	Dec 04, 2014 10:12 Pac...	00:05	100%			

- Component or component process in an application process that did not run successfully

setup database	Dec 04, 2014 10:11 Pac...	03:4...	50%		
Create database	Dec 04, 2014 10:11 Pac...	03:4...	50%		

- Resource that was not deployed successfully

Starting from the Home page:

1. Go to the Applications List.
2. Choose an application.

3. Click the **View** button.

Example:



A list of the application processes for the application appear.

4. To view the Change History of an application process:
 - a. Choose a process.
 - b. Click the **Change History** button.

Example:



The Change History for the application process opens.

5. To view the change history for an object in the application process,
 - a. Click the **View Details** button.

A list of objects in the application process (components and component processes) appears, and the breadcrumb changes to *Applications/View Run*.
 - b. Choose an object.
 - c. Click the **Change History** button to see the change history for the object.

Viewing the Change History From an Application or Environment

You may want to view the Change History for these objects:

- An application in the [Application Visual Editor](#) on page 445
- An application process and process step in the [Applications Visual Editor](#) on page 355
- A component process and process step in the [Component Process Visual Editor](#) on page 448
- An environment in the [Environments Visual Editor](#) on page 450
- Resources in the [Environment Tier](#) on page 451

Application Visual Editor

1. Go to the Applications List.
2. Select an application.

The Applications Visual Editor opens.

- Click the **Menu** button.

Example:

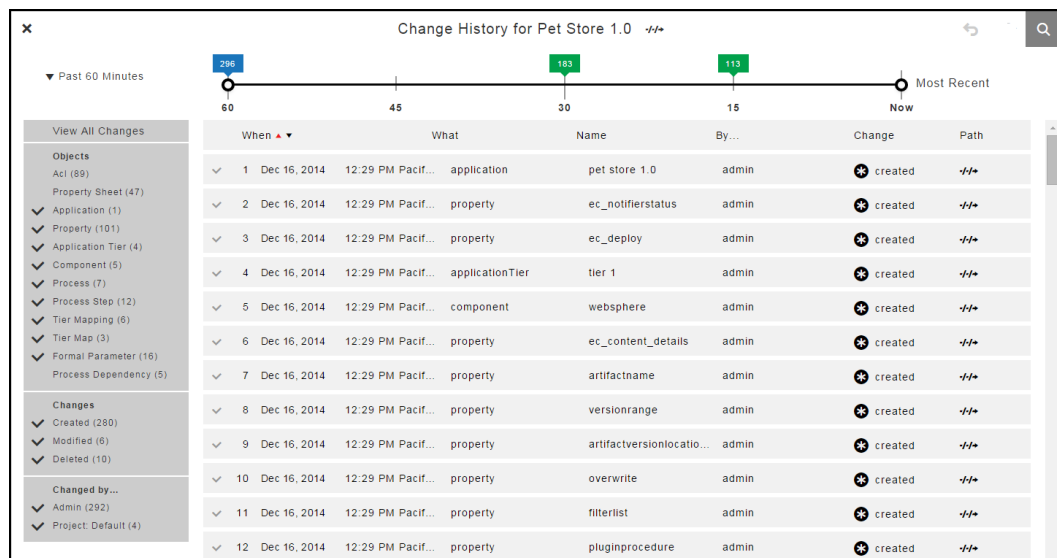


- Select **Track Changes**.

The Change History for the object opens.

The default time increment is **Past 60 Minutes**.

Example:



- Go to the Applications List.

- Select an application.

The Applications Visual Editor opens.

- Select an application process.

The Application Process Visual Editor opens.

4. To view the Change History for the application process:

- a. Click the **Menu** button.

Example:



- b. Select **Track Changes**.

The Change History for the application process opens.

The default time increment is **Past 60 Minutes**.

Example:

Change History for Deploy

▼ Past 60 Minutes

View All Changes	When	What	Name	By...	Change	Path
Objects	1 Dec 16, 2014 12:41 PM Pacif...	process	deploy	admin	created	...
Acl (24)	2 Dec 16, 2014 12:41 PM Pacif...	property	ec_notifierstatus	admin	created	...
Property Sheet (10)	3 Dec 16, 2014 12:41 PM Pacif...	property	ec_deploy	admin	created	...
✓ Process (1)	4 Dec 16, 2014 12:41 PM Pacif...	processStep	s1	admin	created	...
✓ Property (10)	5 Dec 16, 2014 12:41 PM Pacif...	formalParameter	ec_config-version	admin	created	...
✓ Process Step (4)	6 Dec 16, 2014 12:41 PM Pacif...	formalParameter	ec_config-run	admin	created	...
✓ Formal Parameter (9)	7 Dec 16, 2014 12:41 PM Pacif...	formalParameter	ec_smartdeployop...	admin	created	...
Process Dependency (3)	8 Dec 16, 2014 12:41 PM Pacif...	property	ec_notifierstatus	admin	created	...
Changes						
✓ Created (61)						
Changed by...						
✓ Admin (61)						

5. To view the Change History for the application process step:

- a. Choose a step.
- b. Click the **Edit** button in the step to open the context menu.

Example:



- c. Select **Track Changes**.

The Change History for the application process step opens.

The default time increment is **Past 60 Minutes**.

Example:

When	What	Name	By...	Change	Path
1 Dec 16, 2014 12:41 PM	Pacific... processStep	s2	admin	created	...
2 Dec 16, 2014 12:41 PM	Pacific... property	ec_notifierstatus	admin	created	...
3 Dec 16, 2014 12:41 PM	Pacific... property	ec_deploy	admin	created	...

Component Process Visual Editor

1. Go to the Applications List.
2. Select an application.
The Applications Visual Editor opens.
3. Choose a component.
4. Select a component process for the component.

The Component Process Visual Editor opens.

5. To view the Change History for the component process:

- a. Click the **Menu** button.

Example:



- b. Select **Track Changes**.

The Change History for the component process opens.

The default time increment is **Past 60 Minutes**.

Example:

Change History for SetProperties						
<div> <div>▼ Past 60 Minutes</div> <div> <div>9</div> <div>Most Recent</div> </div> </div>						
View All Changes	When ▲ ▼	What	Name	By...	Change	Path
Objects						
Act (5)						
✓ Process (1)						
Property Sheet (3)						
✓ Property (9)						
✓ Process Step (1)						
Changes						
✓ Created (19)						
Changed by...						
✓ Admin (19)						
✓ 1	Dec 16, 2014 12:32 PM Pacif...	process	setproperties	admin	* created	././
✓ 2	Dec 16, 2014 12:33 PM Pacif...	property	repositoryname	admin	* created	././
✓ 3	Dec 16, 2014 12:33 PM Pacif...	property	artifactname	admin	* created	././
✓ 4	Dec 16, 2014 12:33 PM Pacif...	property	artifactversionversion	admin	* created	././
✓ 5	Dec 16, 2014 12:33 PM Pacif...	property	compress	admin	* created	././
✓ 6	Dec 16, 2014 12:33 PM Pacif...	property	dependentartifactv...	admin	* created	././
✓ 7	Dec 16, 2014 12:33 PM Pacif...	property	excludepatterns	admin	* created	././

6. To view the Change History for the component process step:

- a. Choose a step.
- b. Click the **Edit** button in the step to open the context menu.

Example:



- c. Select **Track Changes**.

The Change History for the component process step opens.

Example:

Change History for S1

Past 60 Minutes

15

60

45

30

15

Now

Most Recent

View All Changes

Objects

Property (9)

Acl (3)

Property Sheet (2)

Process Step (1)

Changes

Created (15)

Changed by...

Admin (15)

When	What	Name	By...	Change	Path
1 Dec 16, 2014 12:33 PM Pacif...	property	artifactversionversion	admin	created	
2 Dec 16, 2014 12:33 PM Pacif...	processStep	s1	admin	created	
3 Dec 16, 2014 12:33 PM Pacif...	property	artifactname	admin	created	
4 Dec 16, 2014 12:33 PM Pacif...	property	compress	admin	created	
5 Dec 16, 2014 12:33 PM Pacif...	property	dependentartifactiv...	admin	created	
6 Dec 16, 2014 12:33 PM Pacif...	property	excludepatterns	admin	created	
7 Dec 16, 2014 12:33 PM Pacif...	property	followsymlinks	admin	created	

Environments Visual Editor

1. Go to the Environments List.
2. Select an environment.

The Environments Visual Editor opens.

3. Click the **Menu** button.

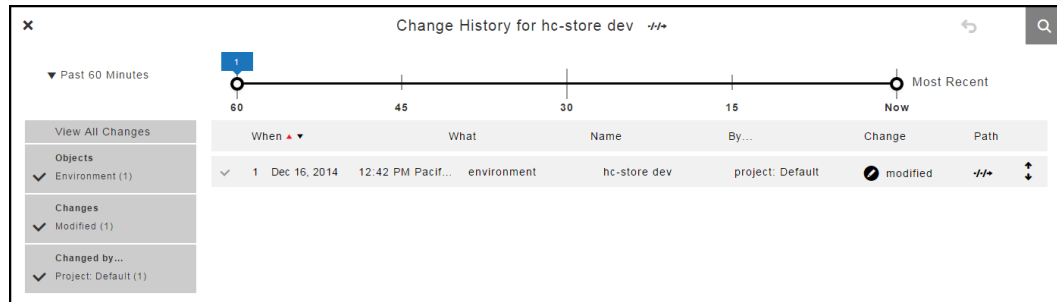
Example:



4. Select **Track Changes**.

The Change History for the object opens.

Example:



Environment Tier

1. Go to the Environments Designer.
2. Choose an environment tier.
3. Click the **Menu** button.

Example:

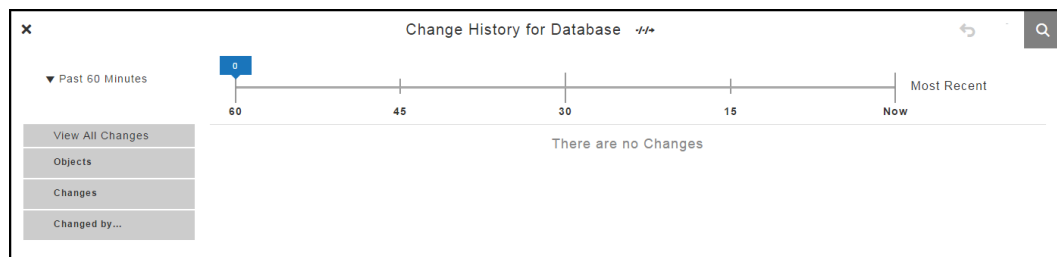


4. Select **Track Changes**.

The Change History for the object opens.

The default time increment is **Past 60 Minutes**.

Example:



Viewing the Change History for Artifacts, Jobs, Projects, and Workflows

When troubleshooting why a job failed, you can view the Change History for artifacts, jobs, projects, and workflows in the automation platform.





- [Artifacts](#) on page 452
- [Jobs](#) on page 453
- [Projects](#) on page 454
- [Workflows](#) on page 454

Artifacts

Starting from the Home page:

1. Go to the Artifacts tab.
2. Choose an artifact.
3. Click the **Track Changes** button.

Example:

Name	Group Id	Artifact Key	Description	Actions
com.ec:myapp	com.ec	myapp		  
com.myccompany.heatclinic.config	com.myccompany.heatclinic	config		  
com.myccompany.heatclinic.warfile	com.myccompany.heatclinic	warfile		  
com.myccompany.heatclinic.warfileWildfire	com.myccompany.heatclinic	warfileWildfire		  

The Change History for the selected artifact opens.

The default time increment is **Past 60 Minutes**.

Example:

When	What	Name	By...	Change	Path
1 Jul 10, 2015 1:02 PM Pacific...	artifact	com.ec:...	admin	created	-/-/

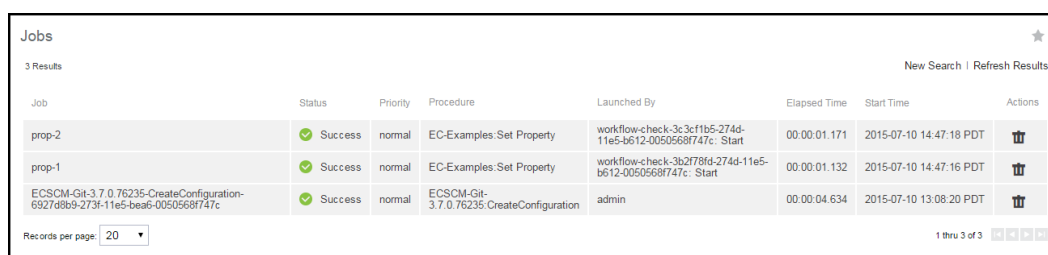
Jobs

Starting from the Home page:

1. To go to the Job Details page, do one of the following:
 - Use the Jobs tab.
 - Use the Jobs Quick View list.
2. If you use the Jobs tab, follow these steps:
 - a. Click the Jobs tab.

The Jobs page opens.

Example:



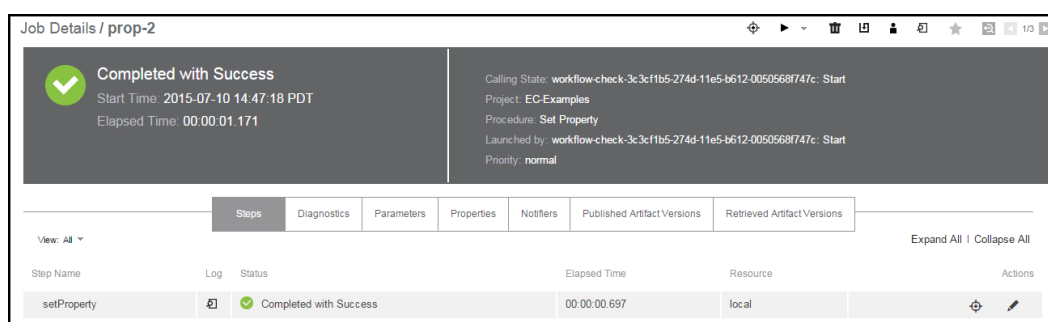
Job	Status	Priority	Procedure	Launched By	Elapsed Time	Start Time	Actions
prop-2	Success	normal	EC-Examples:Set Property	workflow-check-3c3cf1b5-274d-11e5-b612-0050568f747c: Start	00:00:01.171	2015-07-10 14:47:18 PDT	
prop-1	Success	normal	EC-Examples:Set Property	workflow-check-3b278fd-274d-11e5-b612-0050568f747c: Start	00:00:01.132	2015-07-10 14:47:16 PDT	
ECSCM-Git-3.7.0.76235-CreateConfiguration-69278989-273f-11e5-b612-0050568f747c	Success	normal	ECSCM-Git-3.7.0.76235:CreateConfiguration	admin	00:00:04.634	2015-07-10 13:08:20 PDT	

Records per page: 20 1 thru 3 of 3

- b. Click a job name to select a job.

The Job Details page opens.

Example:



Step Name	Log	Status	Elapsed Time	Resource	Actions
setProperty		Completed with Success	00:00:00.697	local	

3. If you use the Jobs Quick View list, click a job name to select a job.

The Job Details page opens.

4. Choose a job or job step.
5. Click **Track Changes** for the job or job step.

The Change History for the job or job step opens.

The default time increment is **Past 60 Minutes**.

Projects

Starting from the Home page:

1. Go to the Projects tab.
2. Choose a project.
3. Click the **Track Changes** button.

The Change History for the project opens.

Example:

Projects						
11 Results					All Projects	Create Project New Search
Project	Description	Impersonation Credential	Create Date	Actions		
Branch_1_2_9	Branch_1_2_9		2015-07-10 15:51:41 PDT			
Default	Default project created during installation.		2015-07-10 12:56:39 PDT			
DemoLibrary	Library of procedures used by the ElectricFlow Demo System.		2015-06-12 00:56:23 PDT			
DeployObjects-1.7.8			2015-07-10 13:02:26 PDT			
EC-Examples	This project contains templates for procedures that perform basic tasks within ElectricCommander.		2015-07-10 12:56:40 PDT			
EC-Tutorials-1.0.0.69904	This project contains small pieces of code which show how to use features in ElectricCommander.		2015-07-10 12:57:51 PDT			
EC-Utilities	This project contains procedures that perform basic tasks within ElectricCommander. Also, the procedures can be used as examples that can be copied and modified in another project.		2015-07-10 12:56:50 PDT			
Electric Cloud	Electric Cloud Procedures		2015-07-10 12:57:18 PDT			
beanstalk	AWS procedures to manipulate Amazon resources for <i>Elastic Beanstalk</i> operations: • S3 storage for file deployments • Elastic Beanstalk Environments • Elastic Beanstalk Applications		2015-01-06 06:47:18 PST			
ec-library	Library of Electric Cloud utility procedures.		2015-01-07 05:55:23 PST			
jpetstore			2015-02-19 05:06:40 PST			

Records per page: 20

1 thru 11 of 11

Workflows

Starting from the Home page:

1. Go to the Workflows tab.
2. Choose a workflow.
3. Click **Track Changes**.

The Change History for the workflow opens.

Change History Page

How to get here: Click the **Search** button to open the "Change History - Search" form, and enter the search criteria.

The search results appear in the Change History page and include the following information:

Change History for JPetStore

▼ Past 60 Minutes

1 2 3 4 5

6 7 8 9

When	What	Name	By...	Change	Path
1 Dec 03, 2014 4:34 PM Pacific...	property	jobcounter	project: ...	modified	...
2 Dec 03, 2014 4:34 PM Pacific...	process	deploy	project: ...	modified	...
3 Dec 03, 2014 12:38 PM Pacific...	property	pluginpr...	admin	created	...
4 Dec 03, 2014 12:38 PM Pacific...	property	pluginpr...	admin	created	...
5 Dec 03, 2014 12:38 PM Pacific...	property	overwrite	admin	created	...
6 Dec 03, 2014 12:38 PM Pacific...	property	directory	admin	created	...
7 Dec 03, 2014 12:38 PM Pacific...	property	type	admin	created	...
8 Dec 03, 2014 12:38 PM Pacific...	property	version	admin	created	...
9 Dec 03, 2014 12:38 PM Pacific...	property	artifact	admin	created	...
10 Dec 03, 2014 12:38 PM Pacific...	property	repository	admin	created	...
11 Dec 03, 2014 12:38 PM Pacific...	property	server	admin	created	...
12 Dec 03, 2014 12:38 PM Pacific...	property...	ec_conte...	admin	created	...
13 Dec 03, 2014 12:38 PM Pacific...	acl	ec_conte...	admin	created	...

How to get here:

From the Home page in the automation platform, use one of the following methods to open and view the change history for jobs, projects, workflows, and artifacts:

- Select a job that failed > Click the job name to go to the Job Details page > Click **Track Changes** in the upper right of the page.
- Click the Jobs tab > Select a job that failed to the Job Details page > Click **Track Changes** in the upper right of the page.

The search results appear in the Change History page for the job and include the following information:

Change History

12/04/14 - 10:45 AM 16 12/04/14 - 6:42 PM

Change History for Proc

1 2 3 4 5 6 7 8 9

View All Changes

Objects

- Process (2)
- Process Step (3)
- Process Dependency (1)
- Act (4)
- Property Sheet (3)
- Property (3)

Changes

- Modified (4)
- Created (12)

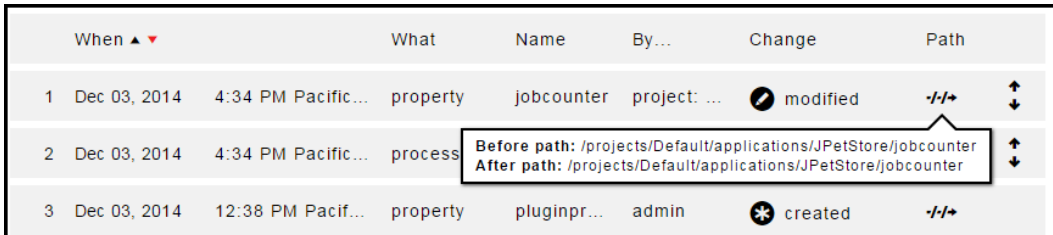
Changed by...

- Project: Default (1)
- Admin (15)

When	What	Name	By...	Change	Path
1 Dec 04, 2014 11:08 AM Pacif...	process	proc	project: Default	modified	/...
2 Dec 04, 2014 11:08 AM Pacif...	processStep	s2	admin	modified	/...
3 Dec 04, 2014 11:07 AM Pacif...	processDepend...	d086ed98-7be8...	admin	created	/...
4 Dec 04, 2014 11:07 AM Pacif...	acl	ec_deploy	admin	created	/...
5 Dec 04, 2014 11:07 AM Pacif...	acl	d3	admin	created	/...
6 Dec 04, 2014 11:07 AM Pacif...	propertySheet	d3	admin	created	/...
7 Dec 04, 2014 11:07 AM Pacif...	propertySheet	ec_deploy	admin	created	/...
8 Dec 04, 2014 11:07 AM Pacif...	propertySheet	d3	admin	created	/...

Close

1	<p>Time range.</p> <p>The end time is the current time. You can change the end time after you run the search and get the search results.</p>
2	<p>Name of the tracked object.</p>
3	<p>Path to the tracked object.</p> <p>Example:</p>
4	<p>After making a change, you can revert or export the object. Click this to go to the undo or redo the last action on the page.</p>
5	<p>Click this to run a new change history search.</p>

6	<p>Time line.</p> <p>The start time is based on the time range that you selected.</p> <p>The end time is the current time.</p> <p>You can manually change the start and end times after you run the search and get the search results.</p>
7	<p>Filters for the change history.</p> <p>You can view all changes or view only selected changes.</p> <p>The objects in the list are the objects in the change history search results.</p>
8	<p>Change history for the selected object.</p> <ul style="list-style-type: none"> • When—the date and time that the object changed. • What—The type of object. • Name—The name of the object. • By—The "user" that changed the object, which can be a project or a user. • Change—The type of change. • Path—Click the View Path button to see the path to the object. 
9	<p>Click the View button to see more details about the object.</p>

Time Line

The time line for the change history is at the top of the Change History page.

The time line is automatically separated into divisions based on the number of minutes, hours, or days between the end time and the start time.

Default Settings

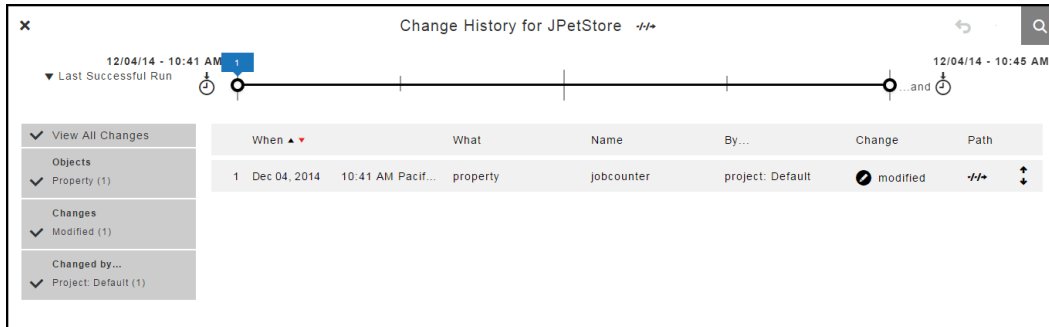
These are the default settings for the time line:

- The start time is the selected time range in the upper left of the page.
- The end time is **Most Recent** when the latest change to the object occurred.

- The default range is from the **Last Successful Run** to the **Most Recent**.
- The entire time line is displayed, and all the changes are in the list below the time line.

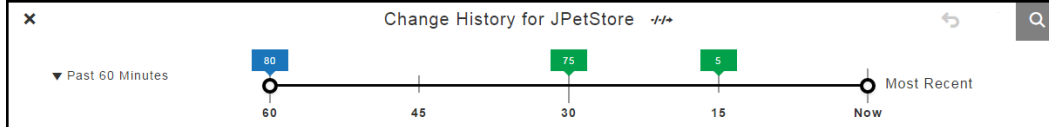
The time line on this page has the default range.

Note: The **Last Successful Run** range is available only after the first time that you run an application.

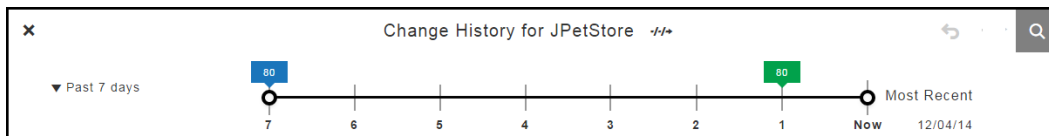


When the range is changed to **Past 60 Minutes**, the time line changes:

- The start time is 60 minutes from **Now**.
- The end time is when the **Most Recent** change occurred (**Now**).
- The time line has four divisions.



If the increment is **Past 7 Days**, the time line has seven one-day divisions.

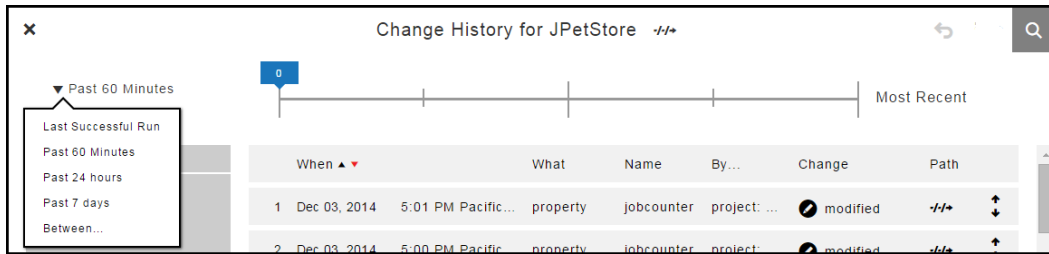


You can view the change history in different ways depending on the time range for the time line.

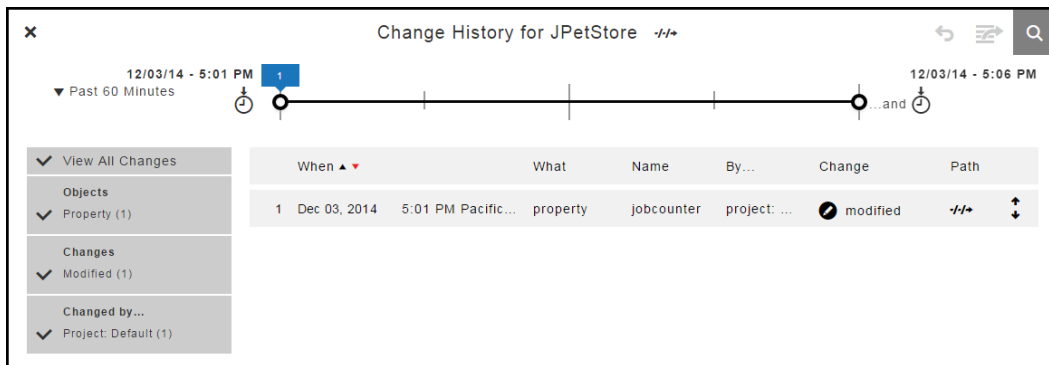
Selecting the Time Range

You can view the change history in different ways depending on the time range for the time line.

Click the down arrow next to the time range to select a different range for the change history.



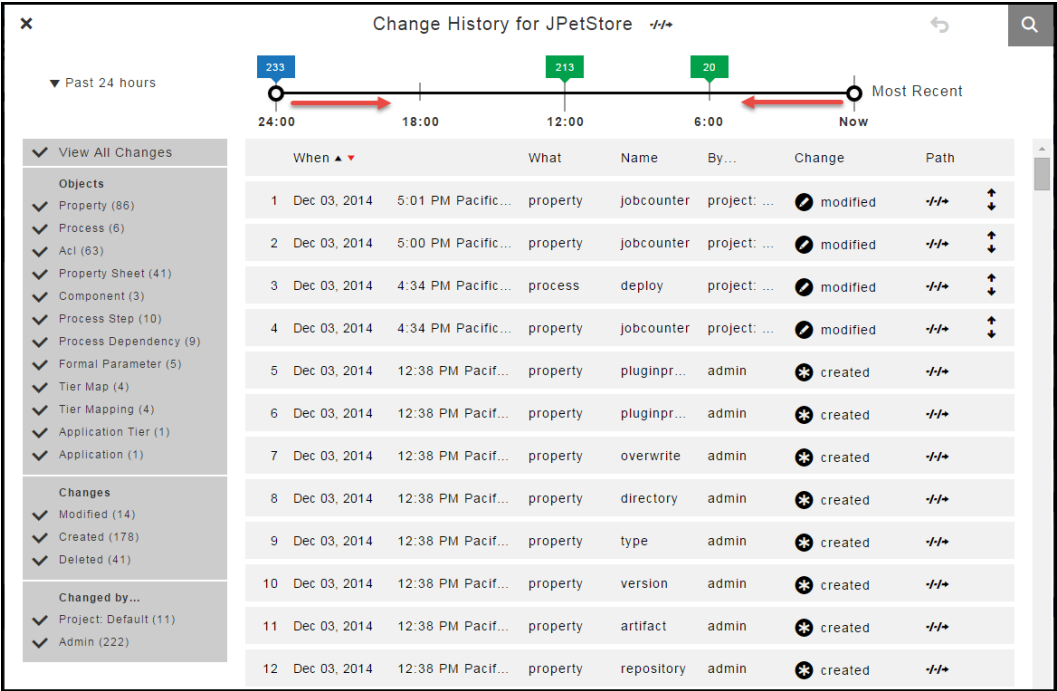
You can see the change history since the last successful run. Notice that the information on the left side shows a summary of the changes, which is a subset of the results that you got.



Moving the Start and End Times

You can manually move the start and end times on the time line.

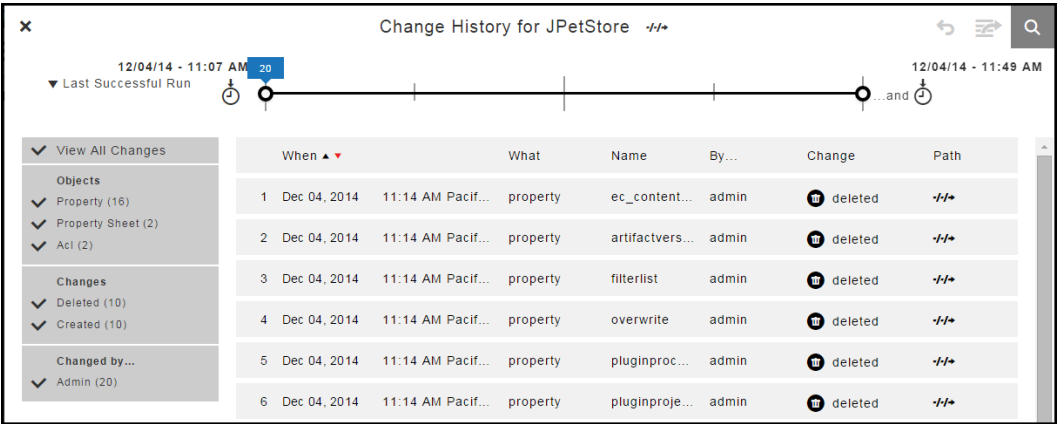
When you move the start time to 18:00 and the end time to 6:00, notice that the list of objects in the change history changes.



When you move the start time to 18:00 and the end time to 6:00, notice that the list of objects in the change history changes.

Specifying the Time Range

Example:



To manually set the times and dates for the start and end times:

1. Select **Between**.

A drop down dialog box opens.

Example:

The screenshot shows a software interface with a sidebar on the left containing the following items: a dropdown menu labeled "Between...", a checked item "View All Char", and three items "Objects", "Changes", and "Changed by...". The main area displays a date and time "11/26/14 - 3:30 PM" next to a blue button with the number "0". Below this is a clock icon. A modal dialog box is open, titled "Time", with input fields for "3", "30", and "PM". Below the time fields is a "Date" section with input fields for "11", "26", and "14". At the bottom of the dialog is a blue "OK" button.

2. Select the time and date for the start of the time line.

The default settings are 3:30 pm and eight days before the current date.

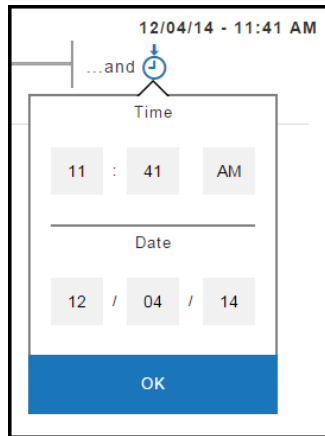
Example:

This screenshot is similar to the one above but shows example values. The main area displays "11/24/14 - 8:30 AM" next to a blue button with the number "0". The modal dialog box, titled "Time", shows "8", "30", and "AM" in its time fields, and "11", "24", and "14" in its date fields. The "OK" button is at the bottom.

3. Click **OK**.

A drop down dialog box opens at the other end of the time line.

Example:

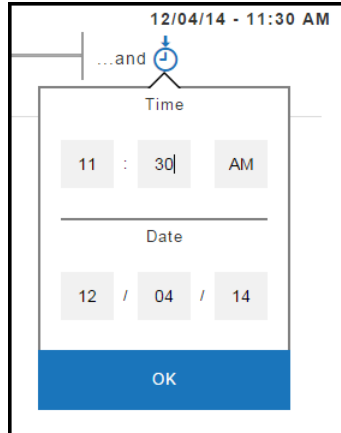


The screenshot shows a dialog box titled "Time" with a date and time selection interface. At the top, it displays "12/04/14 - 11:41 AM". Below this, there is a section for "Time" with input fields for "11", ":", "41", and "AM". Below the time section is a section for "Date" with input fields for "12", "/", "04", "/", and "14". At the bottom of the dialog is a blue button labeled "OK".

4. Select the time and date for the end of the time line.

The defaults are 3:30 pm and the current date.

Example:

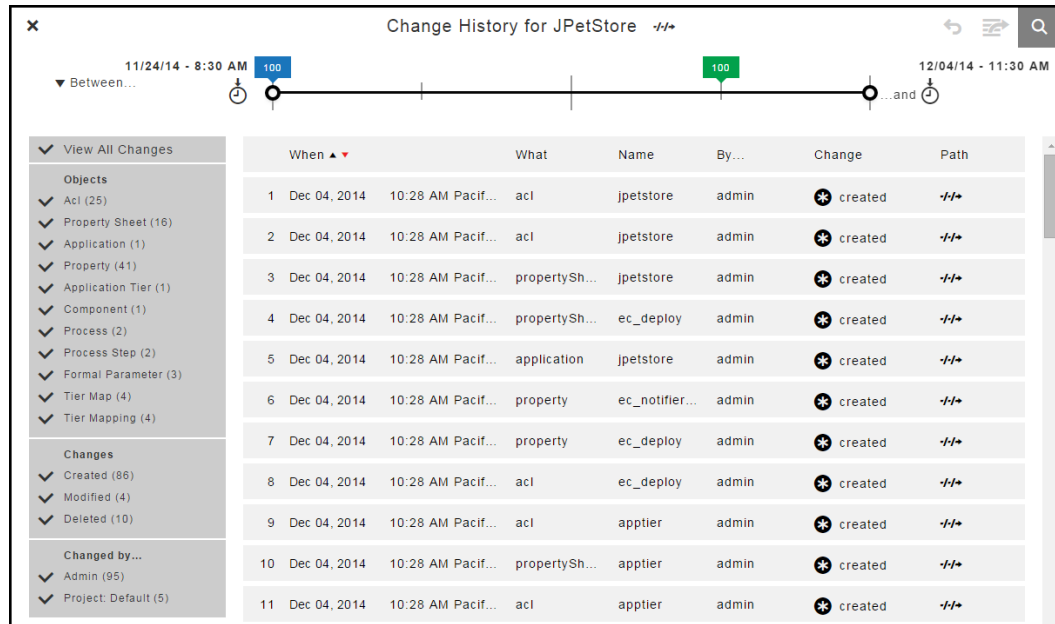


The screenshot shows the same dialog box as before, but with the time changed to "11:30 AM". The date remains "12/04/14". The "OK" button is still at the bottom.

5. Click **OK**.

The time line changes to show only the changes from the start and end times and dates that you selected.

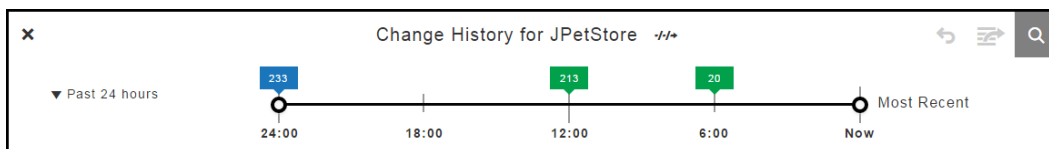
Example:



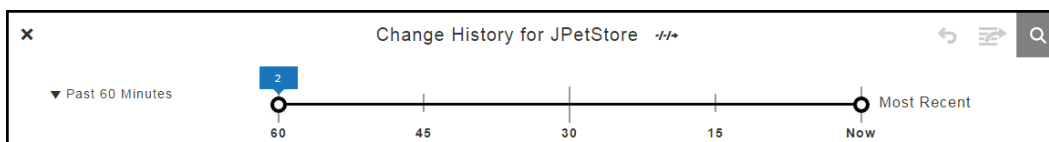
Number of Changes

The time range at the top of the change history shows the number of changes.

- There have been 233 changes in the last 24 hours.
- There have been 213 changes in the last 12 hours.
- There have been 20 changes in the last 6 hours.

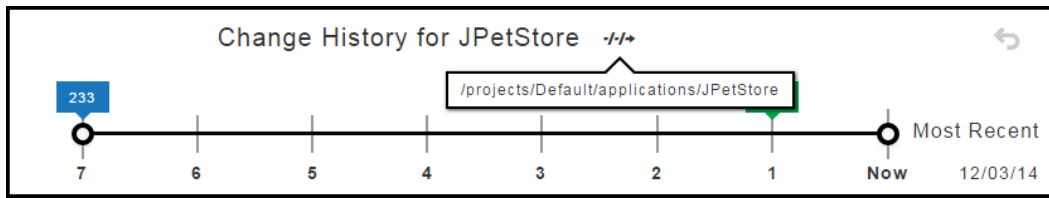


When you change the time range, the number of changes also changes. In the last 60 minutes, there have been only two changes.



Paths to Objects

Click the **View Path** button next to the "Change History for JPetStore" title to see the path to the application.




Click the **View Path** button to see the change in the path to the object before and after the change.

	When ▲ ▼		What	Name	By...	Change	Path	
1	Dec 03, 2014	4:34 PM Pacific...	property	jobcounter	project: ...	modified	↔	↕
2	Dec 03, 2014	4:34 PM Pacific...	process	Before path: /projects/Default/applications/JPetStore/jobcounter After path: /projects/Default/applications/JPetStore/jobcounter				↕
3	Dec 03, 2014	12:38 PM Pacif...	property	pluginpr...	admin	created	↔	

Detailed Object Changes

Click the **View** button to see the change in the property called jobcounter.

When ▲ ▼			What	Name	By...	Change	Path
1	Dec 03, 2014	4:34 PM Pacific...	property	jobcounter	project: ...	 modified	 
Column Changed			From...	To...		Current state	
1. value			5	6		6	

Filters

You can use filters to view changes to specific objects, the types of changes, and the users how made those changes.

Instead of selecting **View All Changes**, you can select specific objects, such as only properties, processes, property sheets, process steps, and process dependencies that have been modified by the Project:Default and Admin users.

View All Changes	When ▲ ▼	What	Name	By...	Change	Path
Objects						
✓ Property (43)	1 Dec 03, 2014 4:34 PM Pacific...	property	jobcounter	project: ...	modified	↕
✓ Process (2)	2 Dec 03, 2014 4:34 PM Pacific...	process	deploy	project: ...	modified	↕
✓ Property Sheet (20)	3 Dec 03, 2014 11:35 AM Pacific...	process	deploy	admin	modified	↕
✓ Acl (27)						
✓ Component (1)						
✓ Process Step (6)						
✓ Process Dependency (8)						
Changes						
✓ Modified (6)						
✓ Created (60)						
✓ Deleted (41)						
Changed by...						
✓ Project: Default (2)						
✓ Admin (105)						

When the list of filter criteria is long, not all of the criteria may appear in the filter list. To see all of the criteria, use the up or down arrows to see all the options.

This list does not show all of the users. Use the up and down arrows to see all four of the users.

✓ View All Changes
Objects
✓ Property (120)
✓ Property Sheet (94)
✓ Process (13)
✓ Tier Mapping (9)
✓ Tier Map (3)
✓ Process Dependency (18)
✓ Acl (144)
✓ Process Step (30)
✓ Component (5)
✓ Application Tier (3)
✓ Application (1)
Changes
✓ Modified (3)
✓ Created (437)
Changed by...
✓ Project: Default (4)
✓ Project: Deploy Objects- ▼

Click the down arrow to see the other users.

The screenshot shows a sidebar with the following sections:

- View All Changes** (checked)
- Objects**
 - ☒ Property (120)
 - ☒ Property Sheet (94)
 - ☒ Process (13)
 - ☒ Tier Mapping (9)
 - ☒ Tier Map (3)
 - ☒ Process Dependency (18)
 - ☒ Acl (144)
 - ☒ Process Step (30)
 - ☒ Component (5)
 - ☒ Application Tier (3)
 - ☒ Application (1)
- Changes**
 - ☒ Modified (3)
 - ☒ Created (437)
- Changed by...**
 - ☒ Project: Deploy Objects-1.7.8 (436) ▲ ▼

Reverting Changes to Objects

You can revert changes that were made to an object and to an objects and its children.

Follow these steps to select the changes that you want to revert:

1. Go to the Change History.
2. Configure the filters to view specific changes in the Change History.

If View All Changes is selected, click it to remove the check mark next to it.

Select only the objects, change types, and the users or groups who made the changes. A check mark appears next to the filter criteria that you select.

While selecting changes, make sure to be aware of the number of changes.

3. Choose an object in the Change History.

- Click the **View** button to view the change details.

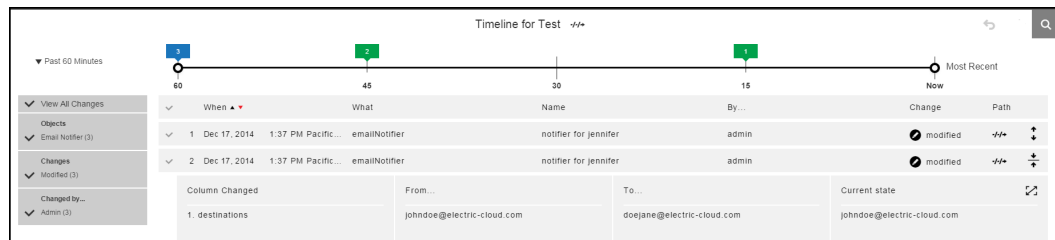
Example:



This information appears:

- Name of the child object (**Column Changed**)
- State of the child object before the change (**From**)
- State of the child object after the change (**To**)
- Current state of the child object (**Current state**).

Example:



- Choose a change to revert.
- In the **To** column, select the row of the object that you want to revert.

Example:

2	Dec 17, 2014 1:37 PM Pacific...	emailNotifier	notifier for jennifer	admin	modified	
Column Changed		From...	To...	Current state		
1_destinations		john.doe@electric-cloud.com	doe.jane@electric-cloud.com	john.doe@electric-cloud.com		

The Revert button is now available (enabled).

Example:



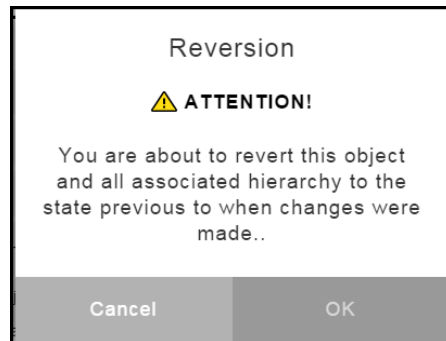
Repeat this step for each change that you want to revert or export.

7. To revert the changes, do the following steps:

- a. Click the Revert button.

A message confirming that you want to revert the selected object appears.

Example:



- b. If you want to revert the selected changes, click **OK**.

If the changes are successfully reverted, a message that the current state of the selected objects are successfully reverted appears.

If the changes are not successfully reverted, a message that the current state of the selected objects are not successfully reverted appears.

- c. If you do not want to revert the selected changes, click **Cancel**.

Credentials and User Impersonation

[What is a Credential?](#) on page 468

[Defining a Credential](#) on page 469

[Why Use Credentials?](#) on page 469

[Credential Access Control](#) on page 469

[Using Credentials for Impersonation](#) on page 470

[Accessing Credentials From a Step](#) on page 472

[Credential References](#) on page 473

[Best Practices for Retrieving Credentials](#) on page 474

What is a Credential?

In ElectricFlow, a "credential" is an object that stores a user name and password for later use.

Two credential types are available, *stored* or *dynamic*:

- **stored** credentials - These credentials are given a name and stored in encrypted form in the database. Each Project has a list of stored credentials it owns. These credentials are managed from the Project Details page.
- **dynamic** credentials - These credentials are captured when a job is created. Dynamic credentials are stored on the server temporarily until the job completes and then discarded.

Defining a Credential

After a credential is created, no one can view the password for the credential's account. This means one person can define a credential and enter the password, and other people can use the credential (and its account) without needing to know the password.

To create a new credential in the automation platform UI:

1. Click the Projects tab.
2. Select a project (first column) to see the Project Details page.
3. Select the Credentials tab and then click **Create Credential**.

Click the **Help** link on the New Credential page if you need more information about what to enter in the fields.

Why Use Credentials?

Use ElectricFlow credentials for:

- User impersonation
- Saving passwords for use inside steps

When a step needs to run as a particular user, ElectricFlow can retrieve the username and password from a stored credential. The credential is passed to the agent over an encrypted channel so the agent can authenticate itself to the operating system and set up a security context where the step runs with the user permissions in the credential.

In some cases, a step needs to enter credentials directly to an application.

- The credential may be a fixed application credential that needs to be presented every time the step runs. In this case, a stored credential provides a place to put this information without needing to embed the password in a script.
- Other times, the password is not known in advance. In this case, a dynamic credential created when a job is launched can prompt a user for information and then pass the credential to the step in a secure manner.

Credential Access Control

Because ElectricFlow stores passwords in a recoverable manner, it is important that credentials are accessible only under carefully controlled circumstances. To protect credentials, ElectricFlow uses two mechanisms: access control lists (ACL) and *attached* credentials.

To use a credential for any purpose, a user who attempts to reference the credential must have execute permission on the credential object—this allows you to explicitly control which users are allowed to use a credential anywhere in the system.

In addition to the user-based access control check, ElectricFlow also requires a credential be explicitly *attached* to the object that is going to use it. After an object has a credential attached, object modifications are restricted to users who have both execute permission on the credential and modify permission on that object. This safeguard helps prevent credentials from accidentally being used by unauthorized users.

Attach a credential to an object in the automation platform UI one of these ways:

- Set the impersonation credential for a step, procedure, project, or schedule.
Impersonation credentials are inherited, so attaching a credential to a container object like a procedure or a project implicitly attaches it to every object inside the container, such as all steps in a procedure.
- Explicitly attach the credential to a step or schedule using the UI or the `attachCredential` API.
To access credentials from a step, the credential must be attached directly to the step, or the credential must be passed as a parameter to the containing procedure and have the parameter attached instead.

Using Credentials for Impersonation

This information addresses the question, "When a job step is running on a resource, under which account is it running and how can I control that?" By default, a job step runs under the account used by the ElectricFlow agent that runs the step, which was chosen when the agent was installed. This approach works well in environments where it makes sense to run all jobs under a single user such as a "build" user. You install all agents to run as the desired user, and you do not have to worry about anything else— every step of every job runs as that user.

However, in other environments you may prefer to run different jobs, or even different job steps, under different accounts.

For example:

- If independent groups are sharing an ElectricFlow system, you may want each group to use a different account for its jobs.
- Or, you may need to run jobs as particular users to access ClearCase views for those users.
- Or, there may be certain steps that require special privileges.

For example, as part of your builds you may need to run a step that generates a certificate using privileged corporate information; that step must run under a special high-privileged account, but you want the remainder of the steps in the build to use a less-privileged account.

ElectricFlow allows you to select accounts on a per-job or per-job-step basis. This mechanism is called *impersonation*, where the ElectricFlow agent impersonates a particular user for the duration of a job step, and this *impersonation* is implemented using credentials. The use of credentials creates special security challenges.

Note: It is important to ensure privileged accounts can be used only for the purposes you intend and cannot be "hijacked" for other purposes. The ElectricFlow access control mechanism contains special facilities to ensure proper credential use.

Setting the Impersonation Credential in the Automation Platform

Attaching Impersonation Credentials to Steps, Procedures, and Projects

You can attach impersonation credentials to procedure steps, procedures, and projects before executing a job step.

ElectricFlow searches for a credential in the following order and uses the first credential it finds:

1. Procedure step
2. Procedure to which the step belongs
3. Project to which the project belongs

If the step is in a nested procedure, and no credential is found on the step, its procedure, or its project, ElectricFlow checks the calling step (and its procedure and project), its caller, and so on until it has worked up through the topmost procedure in the job. If no credential is found, the job step runs under the ElectricFlow agent account on its resource.

This approach makes it easy to manage your account usage. For example, if you want all jobs in one project to use a particular account, define a credential in that project for the account, and then attach the credential to the project. If you want all jobs in a project to use a particular account, except for one step that should use a different account, create two credentials in the project. Attach the first credential to the project and attach the second credential to the particular step.

Attaching Impersonation Credentials to Schedules

You can also attach a credential to a schedule. The credential in the schedule is used for steps where no other credential is available.

The schedule's credential receives the lowest priority.

Attaching Impersonation Credentials to Jobs

You can specify a credential when you launch a job manually one of these ways:

- Provide the name of an existing credential
- Enter an account and password.

If you choose the second approach, the account and password are stored only in ElectricFlow for the duration of the job.

For either way, the credential you specified when you launched the job is used as a last resort for steps only with no other credential.

Attaching a New Impersonation Credential in the Automation Platform

To attach a new impersonation credential for a procedure, step, or schedule,

1. Click the Projects tab.
2. Select a project.
3. Go to the Project Details page for the project you selected.

4. Do one of the following:
 - a. For a **procedure**:
 - i. Click **Create Procedure** to go to the New Procedure page.
 - ii. On this page, select the Impersonation Credential drop-down box to select a credential to use for impersonation.
 - b. For a **step**:
 - i. Select a procedure to go to the Procedure Details page.
 - ii. Select a step name to go to the Edit Step page.
 - iii. On the Edit Step page, scroll down to the Impersonation section.
 - c. For a **schedule**:
 - i. Select the Schedules subtab.
 - ii. Select a schedule to go to the Edit Schedule page.
 - iii. Scroll down to Impersonation Credential section.

For more information in the automation platform UI, each of these web pages contains a **Help** link in the upper-right corner.

Accessing Credentials From a Step

The ElectricFlow server supports an operation called `getFullCredential` that can return a password when called from inside a step. The ectool interface for this operation is:

```
getFullCredential <credential name>  
[--value <password|user>]
```

Without the `--value` option, the response is the same as that returned by `getCredential`, with the addition of a "password" element.

With `--value`, the simple text password or username value is returned on *stdout* so it can be used directly without XML parsing.

The `getFullCredential` API is allowed only inside a running step, and is allowed to use credentials that were explicitly attached to the step only.

Attaching a Credential to a Procedure Step, Procedure, or Schedule

A credential must be explicitly attached to the object using it so the server can perform an access control (ACL) check at definition time and limit the visibility of the password. To support accessing credentials other than the one being used for impersonation, steps, procedures, and schedules contain a list of credential names.

In the automation platform UI, to attach a credential (other than an impersonation credential) to a step, procedure, or schedule:

1. Go to an "Edit" page.

For example, to create a credential for a step, a step must already exist.
2. Select a step that needs a credential. The Edit Step page opens.
3. Scroll to the lower section of the page and click **Attach Credential**.

If you are an ectool command-line user, use the `attachCredential` API command. For more information on the command-line interface, go to the *ectool* Help topics within the online Help system.

Passing Credentials as Parameters

Sometimes you may not know which credential you need to use when you define a procedure step. In this situation, you can leave the credential choice up to the caller of the procedure.

The following example describes the procedure for passing credentials as parameters:

1. Create a project named `InnerProj`.
2. Within that project create a procedure named `InnerProc` with a parameter 'cred' of the type credential.
3. Within that procedure create a step named `InnerStep` with:
 - Command: `ectool getFullCredential cred`
 - Attached Parameter Credential: `cred`
4. Create a project named `OuterProj` with two credentials: `cred1` and `cred2`.
5. Within that project create a procedure named `OuterProc` with two parameters, `cred1` and `cred2`, of the type credential.
6. Within that procedure create two subprocedure steps that both call `InnerStep` from the `InnerProj` project:
 - `OuterStep1` with Attached Parameter Credential `cred1`, and `cred1` in the Parameters section
 - `OuterStep2` with Attached Parameter Credential `cred2`, and `cred2` in the Parameters section

If you are an ectool command-line user, you can use the `createFormalParameter` and `attachParameter` API commands. For more information on the command-line interface, go to the *ectool* Help topics within the online Help system.

Credential References

For schedules, the value for any actual parameter passed to a credential-type formal parameter is interpreted as a reference to a previously attached credential on the schedule.

For procedure steps, the value for any actual parameter passed to a credential type formal parameter is interpreted as a reference to a previously attached credential or credential parameter on the calling step.

When a job step is created, all credential references are resolved and the content is copied into transient credentials on the job step. Credential parameter references are resolved by looking for a credential in the calling job step's transient credential map—looking for a credential with the name specified as the actual parameter value.

Credential references in the API can be specified as a relative or an absolute credential path.

- A *relative* path is similar to "rootCred" and is interpreted as referring to a credential in the current project.
- An *absolute* path includes a project name like `"/projects/MyProject/credentials/rootCred"` and can refer to a credential in any project.

Anywhere a credential reference is accepted (for example, setting an impersonation credential or attaching a credential to a step), either form can be used.

Best Practices for Retrieving Credentials

The ElectricFlow impersonation mechanism provides powerful mechanisms for using different accounts in job steps, and it can be used to handle a variety of challenges. However, if misused, impersonation can open up dangerous security loopholes. This section discusses these risks and how to avoid them.

Use of credentials and impersonation tend to fall into two classes.

First Class

In the first class, your goal is to open up access to accounts, you want to make some accounts generally available, and you trust large groups of users (such as everyone who can access a particular project) to use the accounts appropriately. This usage type is simplest and relatively safe.

1. Define an ElectricFlow group containing all trusted users.
2. Set permissions on the Project so only trusted users can access the project.
3. Define credentials for accounts in the Project and grant execute permission to everyone in the group.

After completing this task, everyone in the group can use the accounts for arbitrary purposes. No one outside the group will be able to access the project or the credentials.

Second Class

In the second class, you want to provide tightly-controlled access to a highly privileged function such as generating a certificate or accessing sensitive information. This kind of usage requires careful thought to avoid allowing unintended access to the credential. The best way to implement a solution for this scenario is to restrict access to the credential and use it on individual steps only after careful analysis of issues such as the following:

Does the step's command use properties?

If so, someone could potentially change the property value to change the step behavior.

For example, suppose the step's command is defined like this:

```
echo $[/myProject/x]
```

Anyone with write access to property x can hijack the step by placing the value "nothing; myCommand" in the property.

After property substitution, the final command will be:

```
echo nothing; myCommand
```

which will cause "myCommand" to execute in the step.

Either restrict access to the property sheet or avoid substitutions.

Does the step execute code or commands that can be modified?

In most jobs that execute build and test sequences, the job extracts various files from a source code control system and some of those files contain code executed during the build. In this case, anyone with access to the source code control system can affect the executed code.

For example, suppose a step runs a Make or Ant command using a configuration file extracted from the source code control system. Anyone with access to the source code control system can modify the configuration file to introduce new commands executed during the Make or Ant step. In this situation, it is virtually impossible to control what happens during the step, so you should never attach a credential for a privileged account to such a step.

The higher you attach a credential, the greater the risk of uncontrolled access to the credential's account.

For example, if you attach a credential to a project, there is a good chance anyone with write access to the project, or even the ability to execute the project's procedures, can hijack the credential using one of the loopholes described above. If you attach a credential to a step that invokes a subprocedure, you effectively give anyone with write access to the subprocedure complete access to the credential's account. If you care about access to an account, you should use its credential in the narrowest possible fashion, such as attaching it to a single job step.

Avoid passing passwords on the command line

When using credentials, avoid passing the password on the command line because [on most platforms] those passwords will be visible to all users logged into the system. The secure way to pass a credential to an application is to use a secured file in the file system or to pipe the value into the application via `stdin`.

Attaching Credentials in Deployment Automation

When modeling an application, you can attach credentials as follows:

- Credentials: Attach one or more credentials to component or application process steps.
- Credentials for impersonation: Attach only one impersonation credential to these objects:
 - Component process
 - Component process step
 - Application process
 - Application process step

When you deploy (run) the application, ElectricFlow applies the credentials based on the user permissions and deploys the application in one or more environments.

Example

This example describes how you can attach impersonation credentials to an application in the main ElectricFlow UI. The application has these credentials:

- Development (dev)
- Quality Engineering (qe)
- Production (prod)

Users have these privileges:

- User A is allowed to deploy the application to build a MySQL database in any environment and has admin privileges.

- User B is allowed to only deploy the application in the development and "quality engineering" environments and is not trusted in the production environment.

The following user permissions determine what users are allowed to do in ElectricFlow. You can configure these credentials only from the main ElectricFlow UI.

- You configure User A's profile in the automation platform and give User A higher-order privileges than other users. User A has the following credentials, including a credential for impersonation:

For each environment in ElectricFlow, set a property using a reference such as `$(myEnvironment/dbConfigName)` and define a unique value, which can be passed as a credential to a process or process step.

- In the development (dev) environment, set `dbConfigName = dbUser_dev`.
 - In the quality engineering (qe) environment, set `dbConfigName = dbUser_qe`.
 - In the production (prod) environment, set `dbConfigName = dbUser_prod`.
- When you configure User B's profile in the automation platform, User B is only given the credentials to deploy in the production environment. You do not need to set properties to be passed as credentials when the application is deployed.

Attaching the same credentials from the automation platform is more complicated. Instead of setting only one credential for User A and one for User B, you need to create three unique credentials for the environments in addition to credentials for the various user and environment combinations, such as User A and the development environment.

Process Branching

This section describes Processing Branching and how to use it.

More about application, deploy, and run:

As you use ElectricFlow, remember that these terms have different meanings within ElectricFlow *and* outside of ElectricFlow when you deploy your software or application:

Term	Within ElectricFlow	Outside of ElectricFlow
Application	The application that you design and run (deploy) to produce your software for continuous delivery across different pipelines.	The software, system or application that you build, test, install, implement, release, and deploy using ElectricFlow. This is the end product of using ElectricFlow.
Deploy	Running the application that you designed in ElectricFlow. The end product is your software, system, or application. Deploy is a synonym of run in ElectricFlow.	All the processes or actions to develop and run your software in its environment, including building, testing, implementing, installing, configuring, making changes, and releasing.
Run	Running the application that you designed. The end product is your software, system, or application. <i>Run</i> is a synonym of <i>deploy</i> in ElectricFlow.	All the processes or actions to use software in its environment, including implementing, installing, configuring, debugging, troubleshooting, and releasing.

About Process Branching

You can use process branching to specify the path through an application or component process based on transition conditions other than out-of-the-box options. Decisions about the next step in the process are made while the process runs. This is similar to the transition conditions for workflows in the automation platform.

If the application or component process applies to multiple use cases, you can design one process with two or more branches instead of designing multiple processes for each use case. You can also define steps that run in parallel.

For example, to install or upgrade software, you can define one process for multiple use cases and use the same steps except for the following:

- The source files can be in .zip or .tar format. The steps to extract the files depend on the format.
- The operating system can be Linux or Windows. The steps to download the files, install them on the server, and enter commands depend on the operating system.

ElectricFlow supports the following branching conditions. The default is **Always**.

- Completion status of the previous process step
- A property set in another part of the system, not the in the previous step
- Custom validation rules

When you define a step in an application or component process, you configure what ElectricFlow does when an error occurs. You select **stop running** or **continue running** in the On Error field in the Define Step dialog box.

This setting overrides any job-step-level branching condition. If an error occurs in a job step and the **stop running** is set, ElectricFlow aborts even if the branching condition is fail.

How to Use the Process Branching UI

How to get to the Application Process Visual Editor:

- Existing application process: From the Applications Visual Editor, click the down-arrow button and select an application.

The Application Process Visual Editor for that application process appears.

- New application process: From the Applications Visual Editor, click the **Add application process** button, set the parameters in the **Application Process Details** dialog box, and click **OK**.

The Application Process Visual Editor for the application process appears.

How to get to the Component Process Visual Editor:

- Existing component process: From the Applications Visual Editor, click the **Number of component processes** button, and select a component process in the drop-down list.

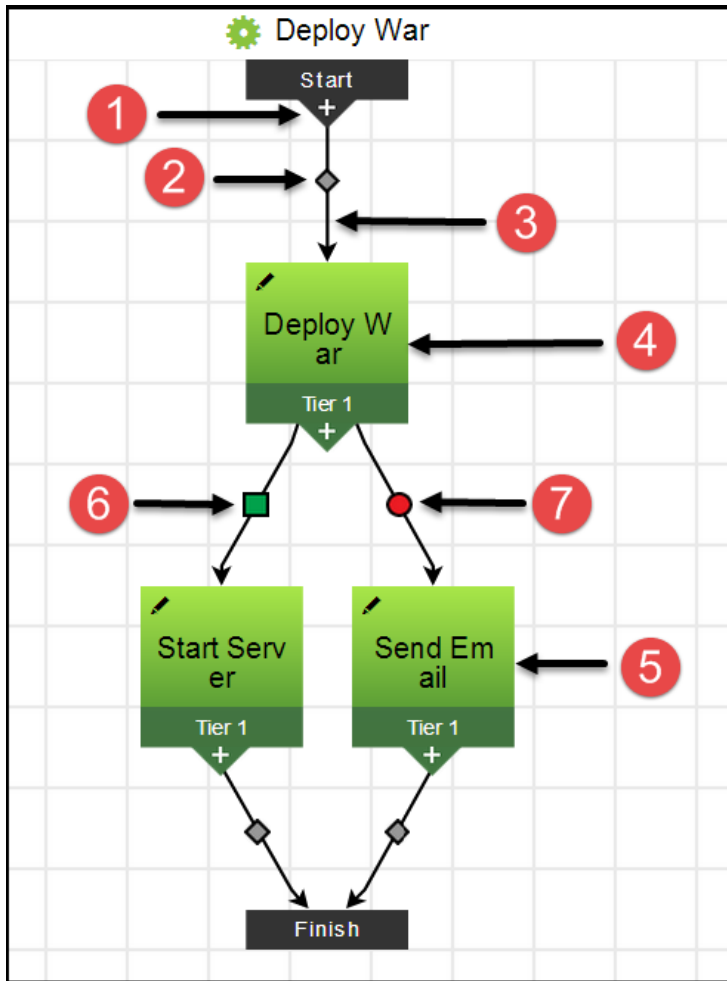
The Component Process Visual Editor for that component process appears.

- New component process: From the Applications Visual Editor, click the **Add component process** button to a component, set the parameters in the **Component Process Details** dialog box, and click **OK**.

The Component Process Visual Editor for the component process appears.

UI Objects

This example shows how a process appears in the ElectricFlow UI.



The process has these UI objects:

1	<p>Click the plus sign (+) to add a step after the selected step.</p> <p>In this example:</p> <ul style="list-style-type: none"> • When you click the plus sign, a new step is added after the Start and parallel to the existing next step called Deploy War. • The steps immediately after the "Deploy War" step are parallel steps. The decision about the next step, either the "Start Server" or the "Send Email" step, depends on the branching condition. The results of the "Deploy War" step determine what the next step is. <ul style="list-style-type: none"> ◦ If the results of the "Deploy War" step are successful (shown by the square green connector), the next step is the "Start Server" step. ◦ If the results fail (shown by the circular red connector), the next step is the "Send Email" step.
---	---

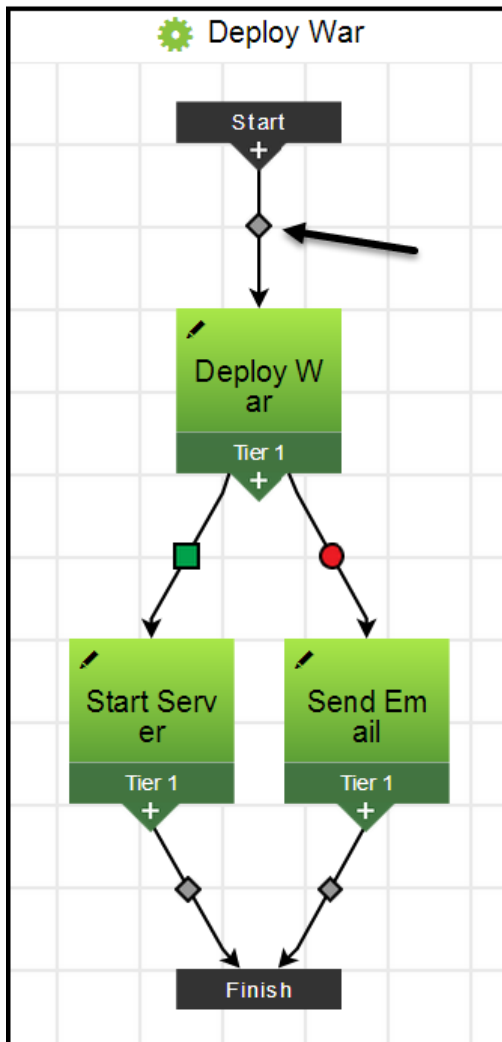
2	Connector between two objects in the process. The default branching condition is Always. When you click the connector, the branching conditions menu opens.
3	Link between two steps in the process. The link goes from the source step to the target step.
4	Source—The link starts at this step. For the link between the "Deploy War" and the "Send Email" steps, the source is the "Deploy War" step.
5	Target—The link ends at this step. For the link between the "Deploy War" and the "Send Email" steps, the source is the "Send Email" step.
6	The branching condition is Successful . If the War file is run successfully in this example, the next step is Start Server.
7	The branching condition is Failure . If the file is not run in this example, the application fails and the next step is to send an email to the administrator.

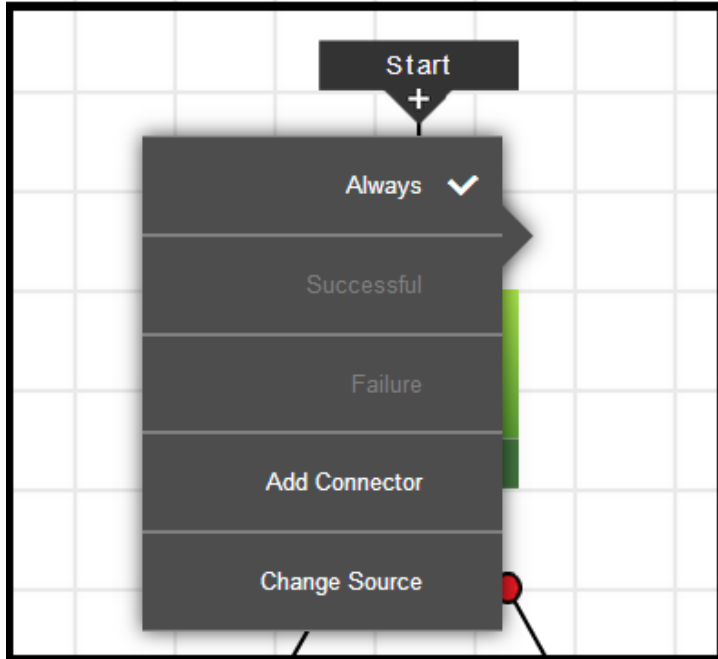
Branching Conditions Menu

When you click a connector on a link, the branching conditions menu opens. Depending on the location of the connector, some of the menu options may not be available. These are possible branching conditions:

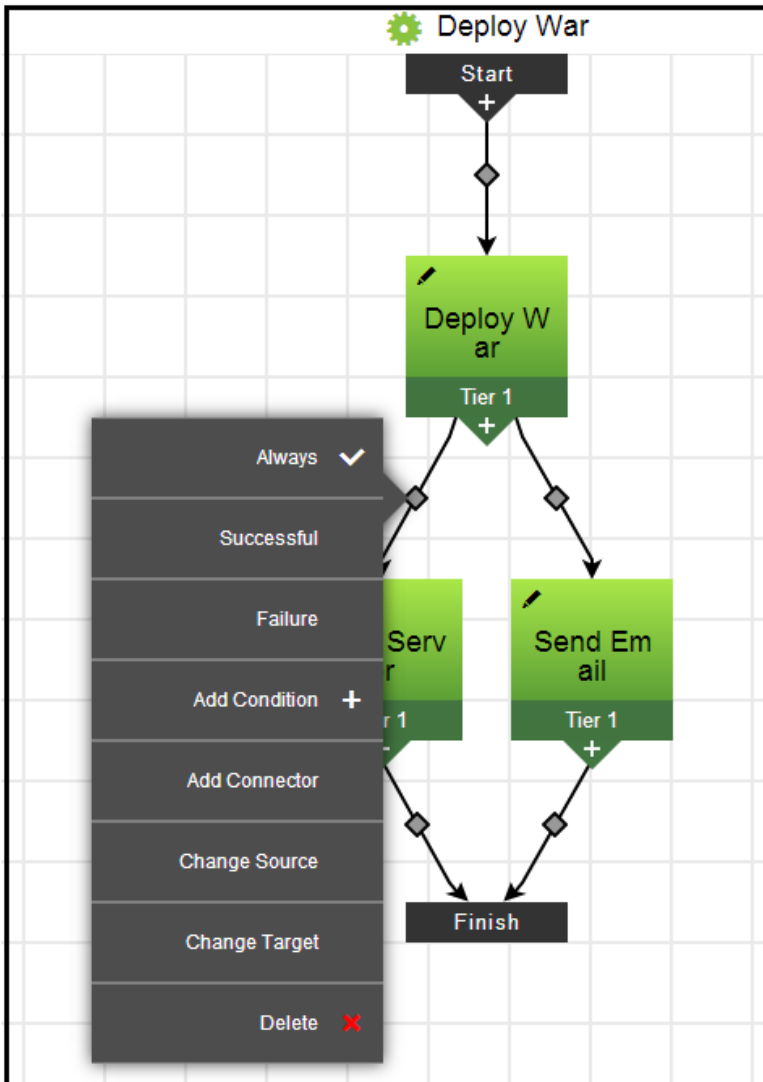
- **Always**—Always go to the next step, referred to as the target.
- **Successful**—Go to the next step if the previous step, referred to as the source, is successful.
- **Failure**—Go to the next step if the previous step fails.
- **Add Condition**—Add a custom condition.
- **Add Connector**—Add a connector from the source of the link to a new target by selecting one of the highlighted eligible steps. You can only select an eligible step.
- **Change Source**—Change the source by selecting one of the highlighted eligible steps, which has a red outline. You can only select an eligible step.
- **Change Target**—Change the target by selecting one of the highlighted eligible steps, which has a red outline. You can only select an eligible step.
- **Delete**—Delete the selected connector and link.

For example, when you select the connector between the Start and "Deploy War" steps, only some of conditions appear and only some are available. The condition between the Start and the next step is **Always**, the default branching condition.



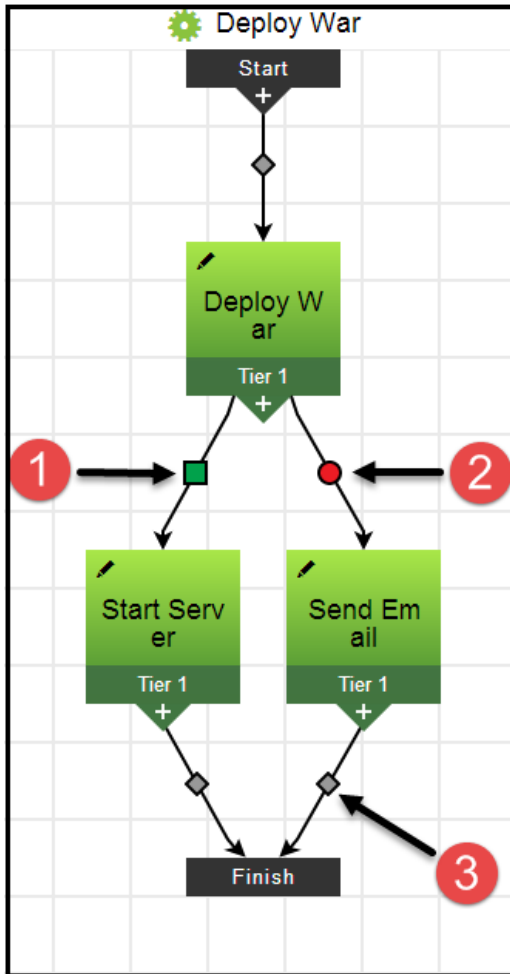


When you select the connector between the "Deploy War" and "Start Server" steps, all of the these conditions appear.



Analysis of a Process With Simple Branching

This example shows a process with simple branching that runs the War file. For the steps to design this example, see [Simple Process Branching Example](#) on page 490.



1	The branching condition is Successful . If the War file is run successfully, the next step is Start Server.
2	The branching condition is Failure . If the file is not run successfully, the application fails and the next step is to send an email to the administrator.
3	The branching condition is Always .

The example has the following job-step branching dependencies:

Source	Target	Branching Condition
Deploy War	Start Server	Successful

Source	Target	Branching Condition
Deploy War	Send Email	Failure

Process Branching States and Conditions

State of the Branching Condition Connectors in the UI

In the ElectricFlow UI, the status of the link is based on the shape and color of the connector.

Shape	Color	Link Status
Diamond	Light gray	Always
Diamond	Dark gray	Disabled
Square	Green	Successful
Circle	Red	Failure

Examples of Branching Conditions

These are examples of branching conditions that you can apply in your processes.

- Based on the status of the previous step

Follow the branch based on the result of the previous step: Successful, Failure, or both (Always).

Example:

- **Successful**—If the file is downloaded successfully, the next step is to extract the files.
- **Failure**— If the file was not downloaded properly, the next step is to abort the process.
- **Always**—The next step is to always extract the files.

- Based on a value of an operation during the step

Follow the branch that matches the result of an operation such as calculating a value or processing data during the step.

Example: The result of an operation is a file type.

- If the result is an XML zip file, the next step is open an XML text editor.
- If the result is a .htm file, open a web browser.
- If the result is a .mov, open an application to play the movie.

- Based on a property in another part of the system

Follow the branch based on a property set in another part of the system, not in the previous process step.

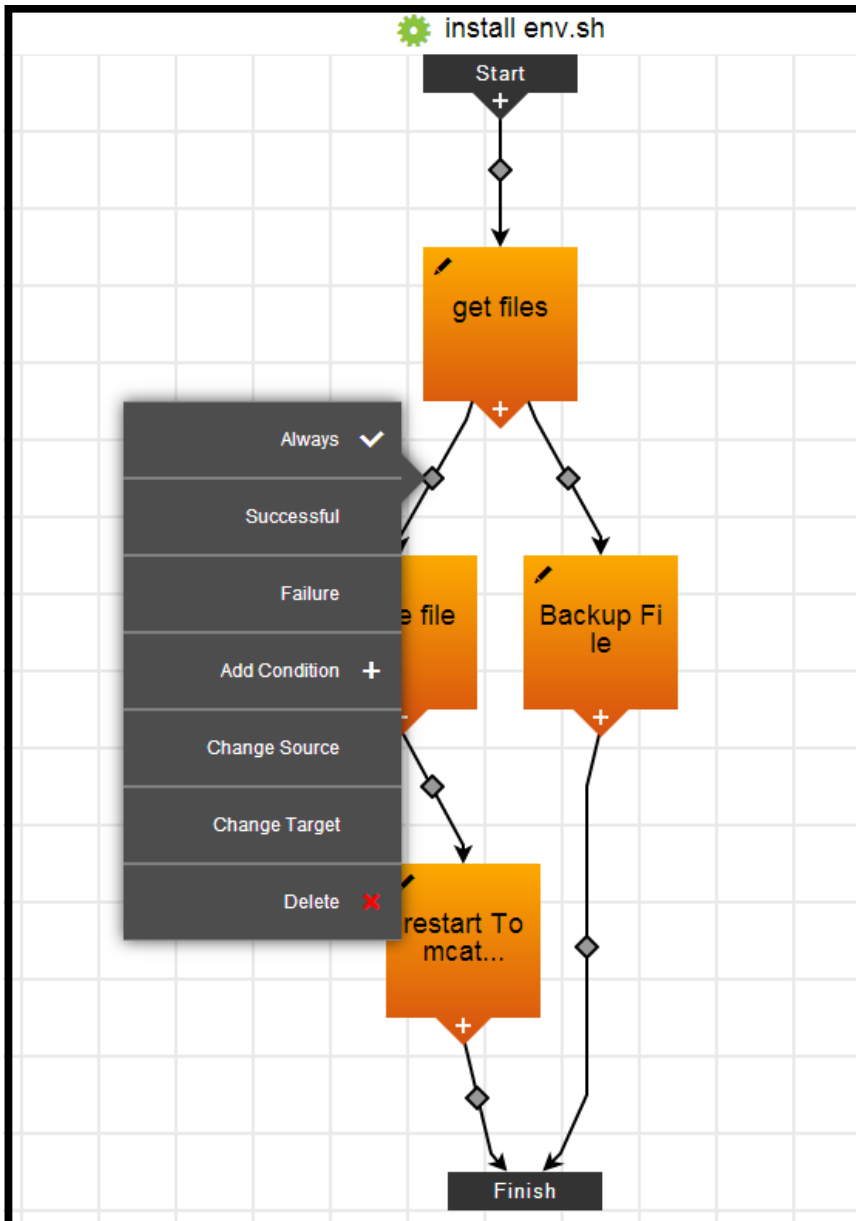
Example:

- If the property `os_type = linux` is set on a resource, always follow the branch for Linux steps.
- If the property `release_type` is set to minor in the application, always follow the branch for minor releases when running the process.

Custom Conditions in Process Branching

How to get here: In a component or application process with branching, click the connector on a link to open the branching options menu.

In this example, click the connector between the "Deploy War" and "Start Server" steps, and select **Add Condition** to add a custom condition.



The Condition dialog box opens.

Condition Delete

[Condition Name]

Property based... ☒ Custom ☐

on... Select Object...

Property

Select... >

is... Select Condition...

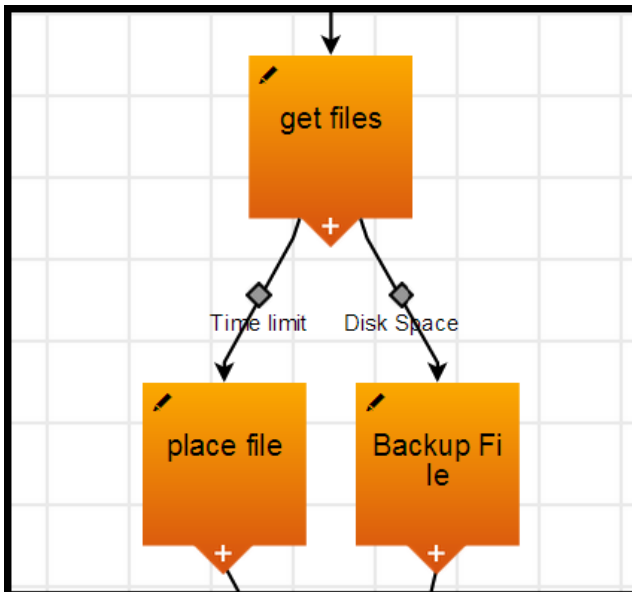
<Value>

Cancel OK

When you click the connector between the "Deploy War" and "Start Server" steps, these conditions appear:

- **Always**—Always go to the next step, referred to as the target.
- **Successful**—Go to the next step if the previous step, referred to as the source, is successful.
- **Failure**—Go to the next step if the previous step fails.
- **Add Condition**—Add a custom condition.
- **Add Connector**—Add a connector from the source of the link to a new target by selecting one of the highlighted eligible steps. You can only select an eligible step.
- **Change Source**—Change the source by selecting one of the highlighted eligible steps, which has a red outline. You can only select an eligible step.
- **Change Target**—Change the target by selecting one of the highlighted eligible steps, which has a red outline. You can only select an eligible step.
- **Delete**—Delete the selected connector and link.

After you configure your conditions, they appear near the affected connectors in the process.



When you configure a **Property based** condition, the fields in the Condition dialog box remain the same .

When you configure a **Custom** condition, the fields change.

The screenshot shows a dialog box titled 'Condition' with a close button (X) in the top-left corner and a 'Delete' button with an X icon in the top-right corner. Inside the dialog, there is a text input field containing '[Condition Name]'. Below this, there are two radio buttons: 'Property based...' (which is unselected) and 'Custom' (which is selected). Under the 'Custom' radio button, there is a large, empty rectangular area for configuration. At the bottom of the dialog, there are 'Cancel' and 'OK' buttons.

Configuration Guidelines for Process Branching

Follow these guidelines when you use process branching in your application or component processes.

- When you add a step, you must define it before adding another step.
- You can only configure branching conditions on a connector between two process steps.
- You cannot configure branching conditions between these objects:
 - The start of the process and the steps immediately after it.
 - The end of the process and the steps immediately before it.
- You cannot configure branching conditions between these objects:
 - The start of the process and the steps immediately after it.
 - The end of the process and the steps immediately before it.
- When you define a step in an application or component process, you configure what ElectricFlow does when an error occurs.

Select **stop running** or **continue running** in the **On Error** field in the Define Step dialog box. *This setting overrides any job-step-level branching condition.*

If an error occurs in a job step and the **stop running** is set, ElectricFlow aborts even if the branching condition is set to Failure.

Simple Process Branching Example

How to get to the Application Process Visual Editor:

- Existing application process: From the Applications Visual Editor, click the number-and-down-arrow button and select an application. The Application Process Visual Editor for that application process appears.
- New application process: From the Applications Visual Editor, click the **Add application process** button, set the parameters in the **Application Process Details** dialog box, and click **OK**. The Application Process Visual Editor for the application appears.

How to get to the Component Process Visual Editor:

- Existing component process: From the Applications Visual Editor, click the **Number of component process** button, and select a component process in the drop-down list. The Component Process Visual Editor for that component process appears.
- New component process: From the Applications Visual Editor, click the **Add component process** button to a component, set the parameters in the **Component Process Details** dialog box, and click **OK**. The Component Process Visual Editor for the component process appears.

This example shows how to design a new process and run the War file.

- If the application succeeds, ElectricFlow starts the server.
- If the application fails, ElectricFlow sends an email to the administrator.

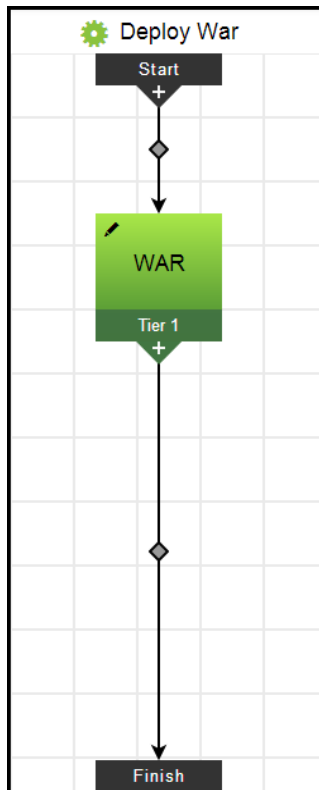
Beginning in the Application Process or Component Process Visual Editor:

1. Click the button below "Define this Step."

The process step dialog box appears.

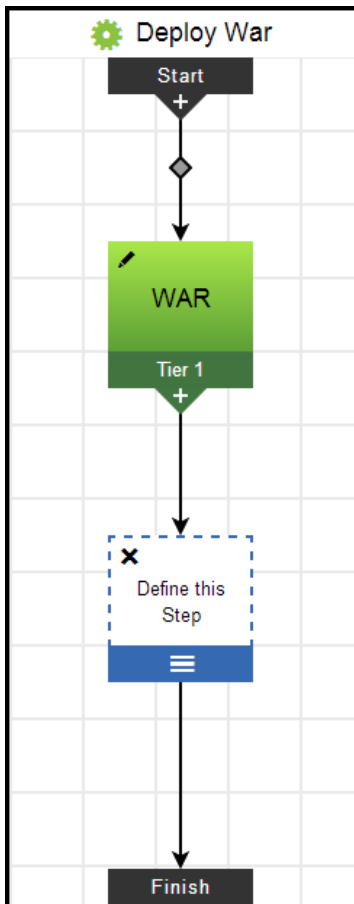
2. Configure the step.

The first step is now configured.

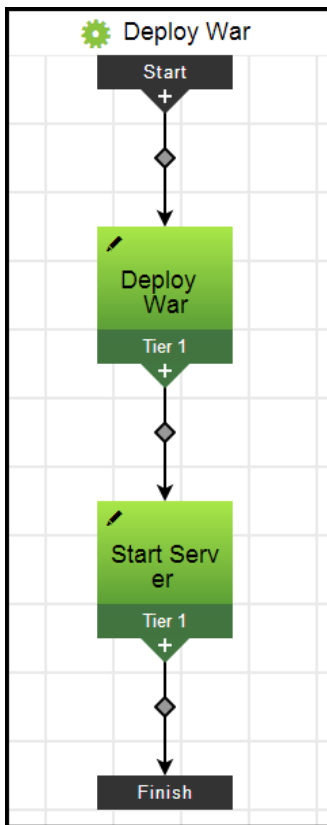


3. To add a step below the first step, click the plus sign (+) below the tier name in the first step.

A new undefined step appears below the first step.

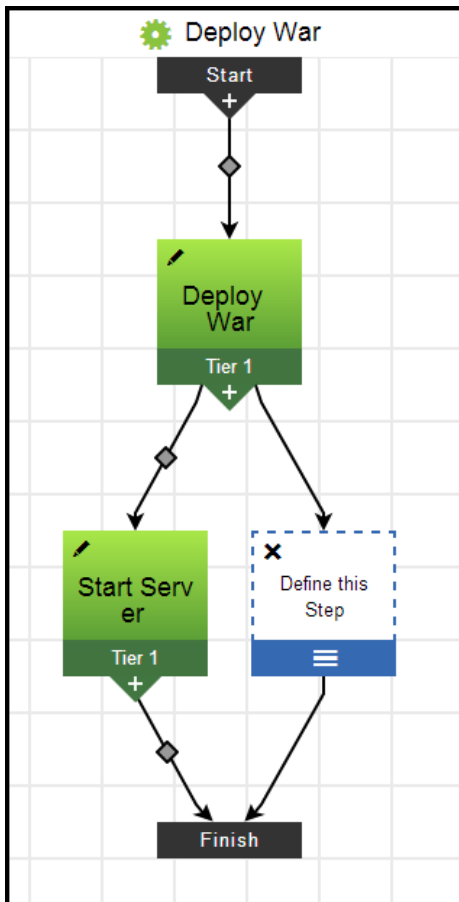


4. Define the new step.

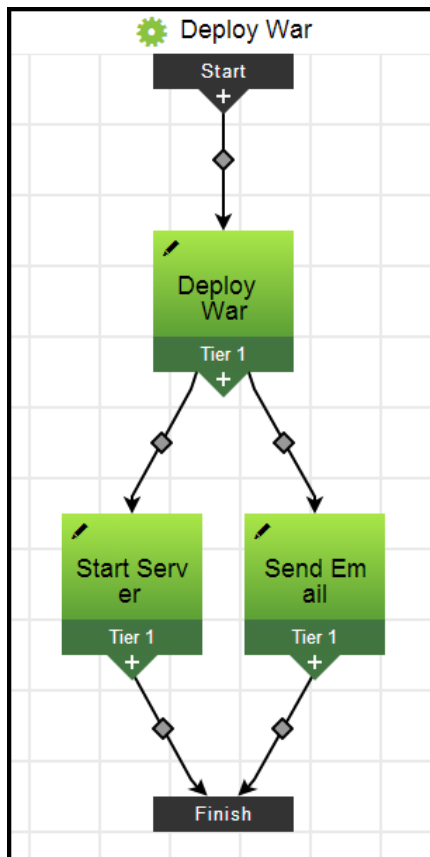


5. To add a step that will be parallel to the second step, click the plus sign in the first step.

A new undefined step appears below the first step and parallel to the second step.

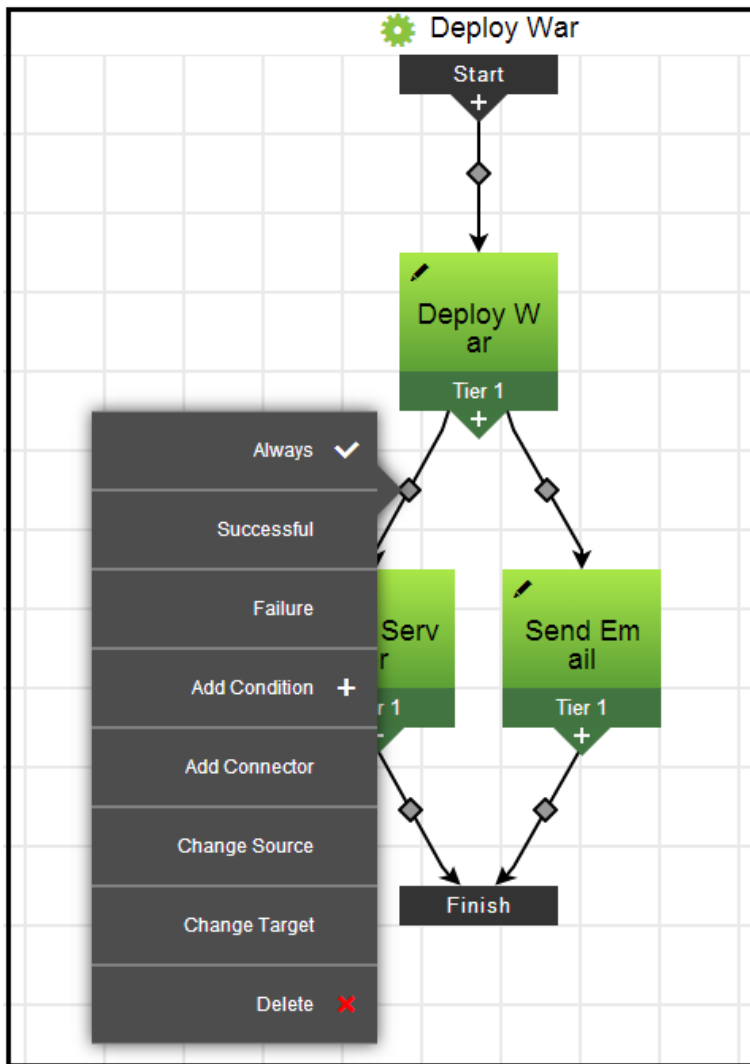


6. Define the third step.



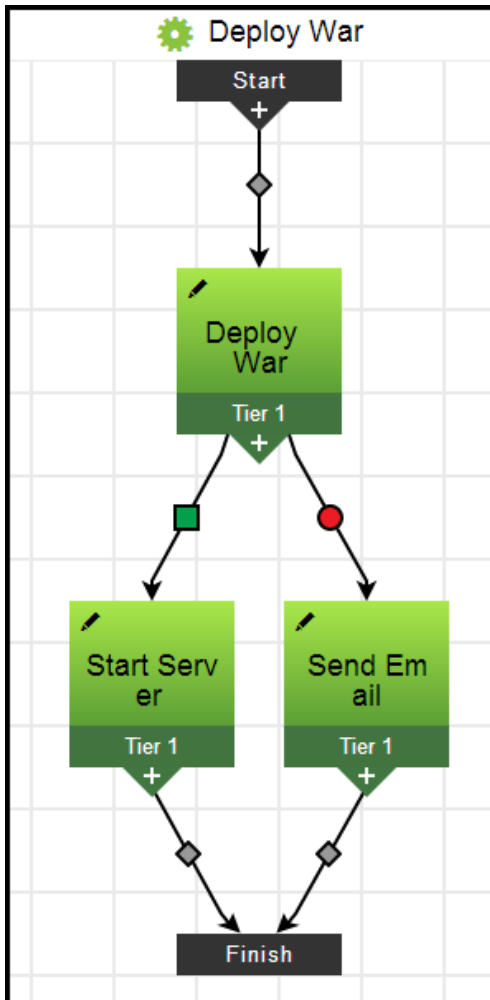
7. To configure the branching condition between the "Deploy War" step and the "Start Server" step, click the connector between them.

The branching conditions menu opens.



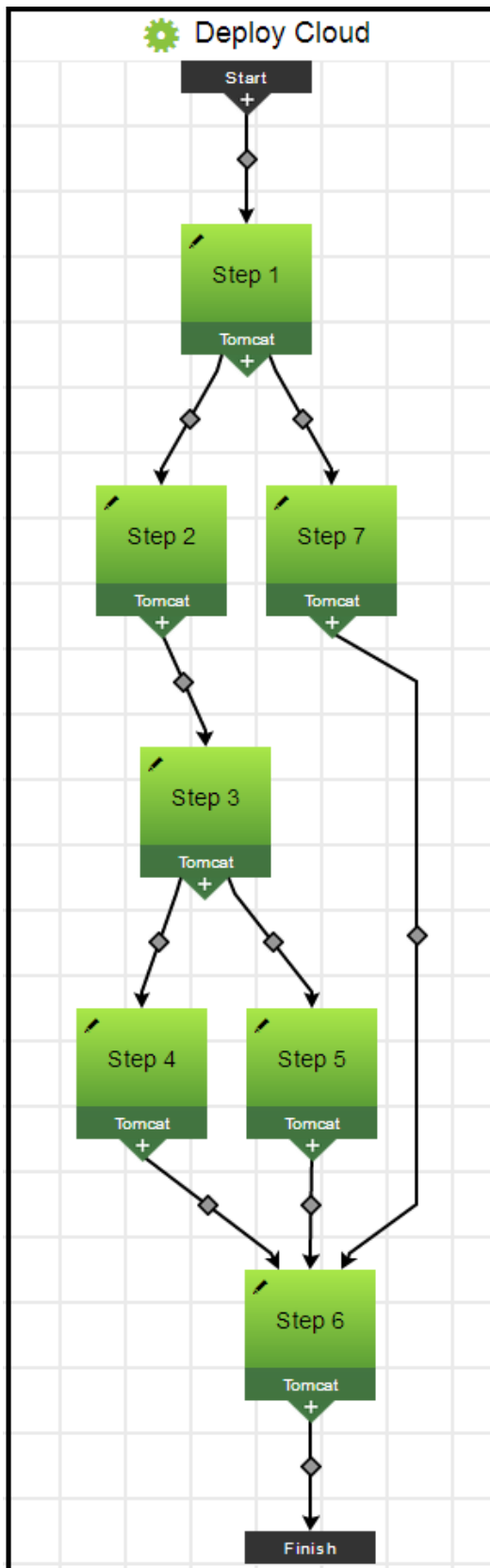
For the "Start Server" step, select **Successful**.

8. To configure the branching condition between the "Deploy War" step and the "Send Email" step, click the connector between them, and select **Failure** in the branching conditions menu.



Process Branching Example: Deleting Steps

This example shows how a process changes when you delete process steps.



Deleting Step 3

To delete Step 3:

1. Click the **Edit** button on Step 3.

The step details menu appears.

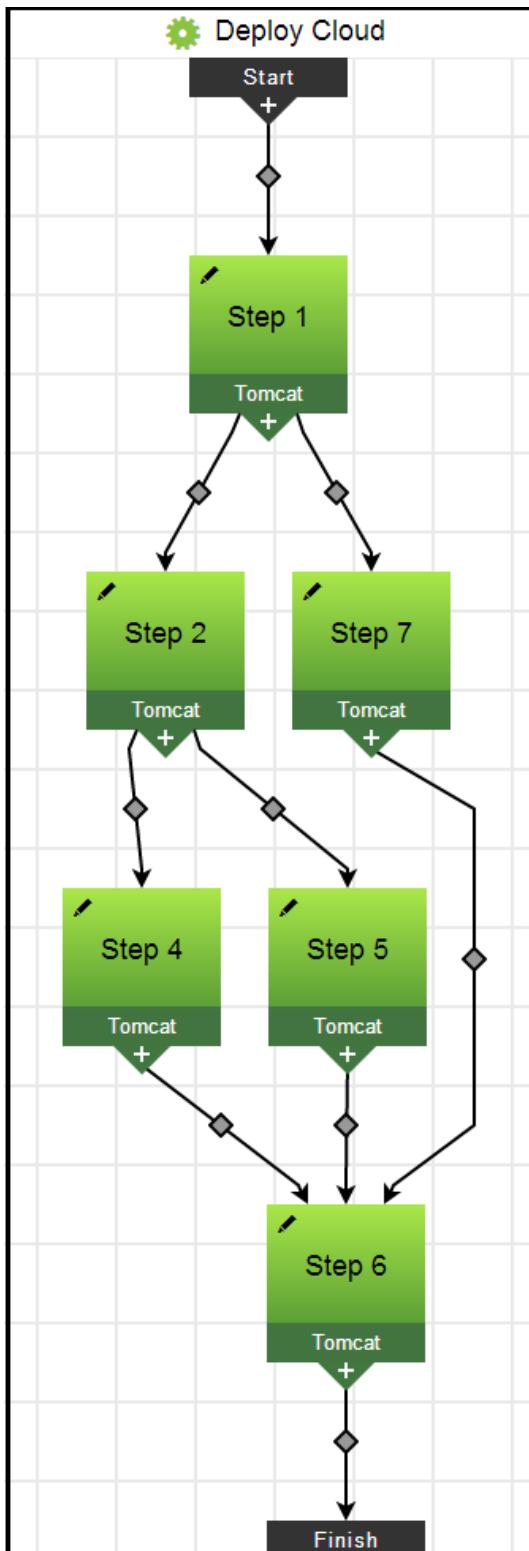
2. Click **Delete**.

The Step Deletion dialog box appears.



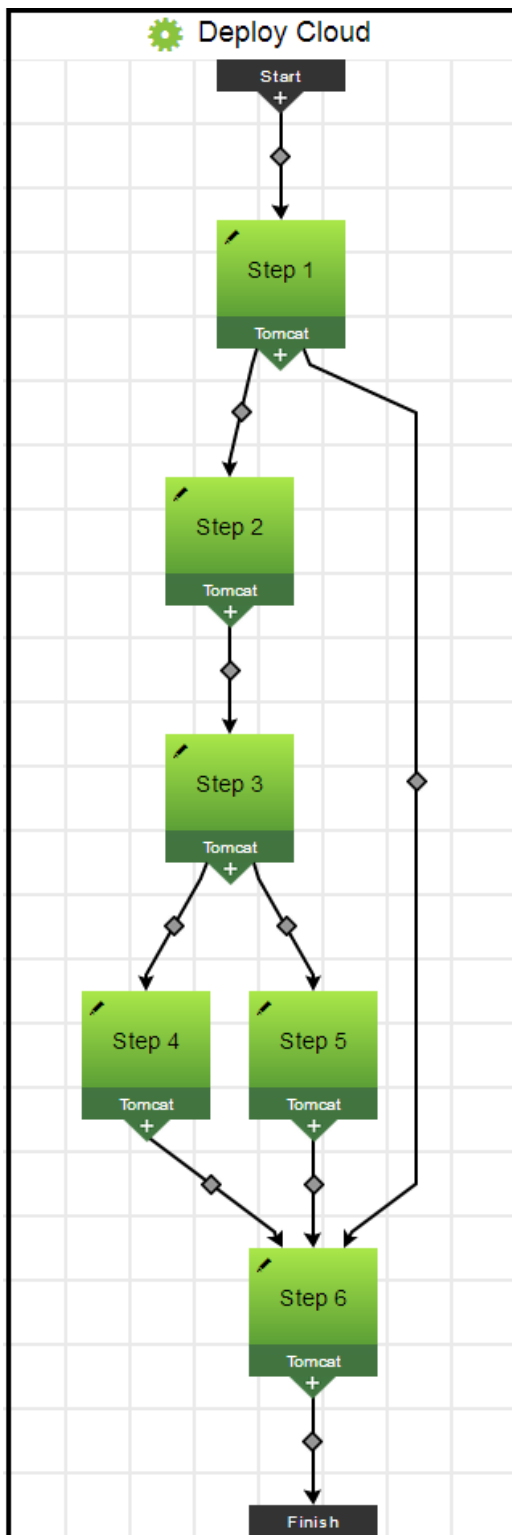
3. Click **OK** to delete the step.

When you delete Step 3, Step 2 becomes the source for Step 4 and Step 5.



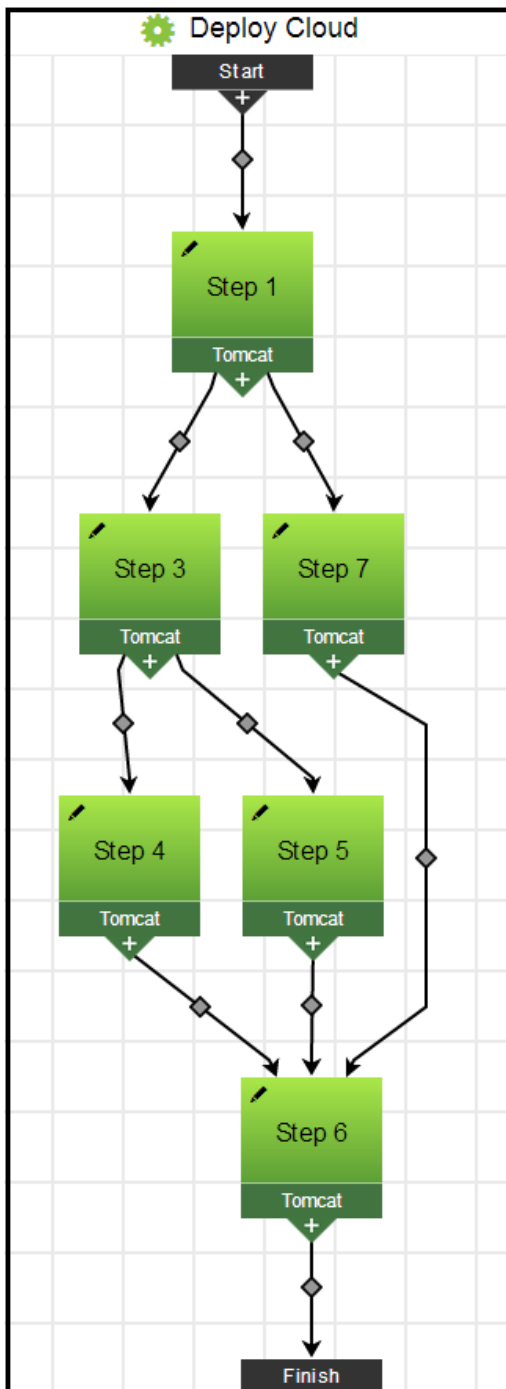
Deleting Step 7

Step 1 becomes the source for Step 6.



Deleting Step 2

Step 1 becomes the source for Step 3.



Adding Snapshots

Follow these steps to add snapshots:

1. Go to the Applications List.
2. Choose an application.
3. Click the **Snapshot** button.

Example:



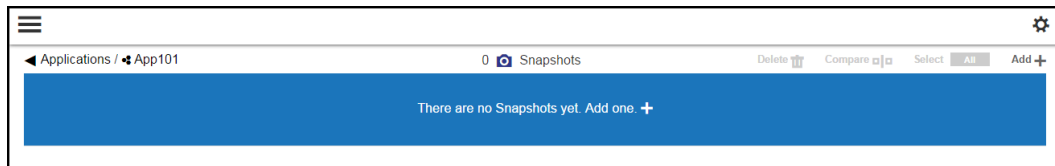
4. Select **Snapshot List**.

The Snapshot List appears.

5. To add a new snapshot, do one of the following:

If there are no snapshots in the list, click anywhere in the **Add one. +** pane to open the New Snapshot dialog box.

Example:

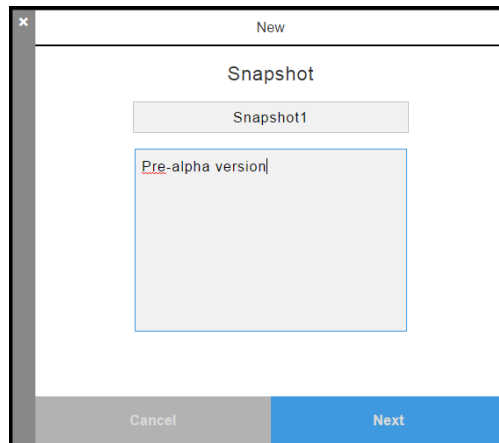


If the list has one or more snapshots, click **Add +**.

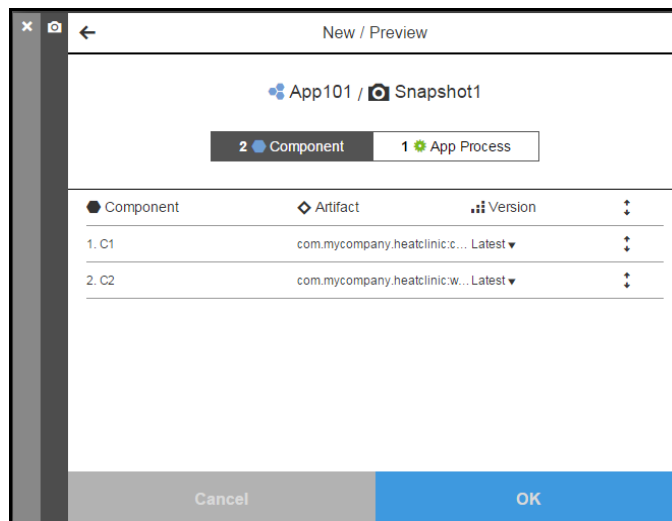
The New Snapshot dialog box appears.

6. Enter a name for the snapshot that must be unique within the application.

7. (Optional) Enter a description of the snapshot.

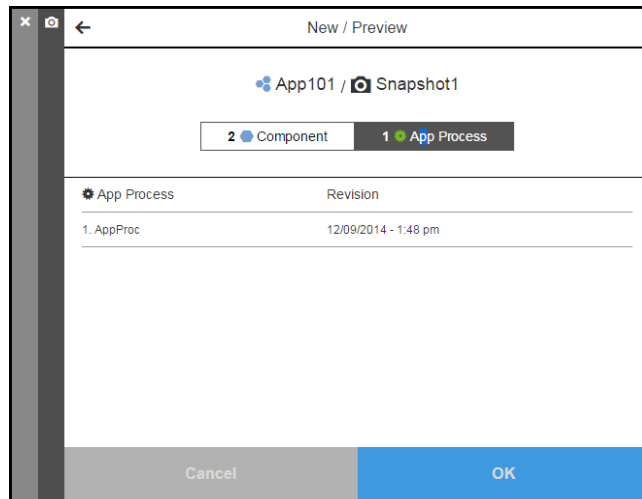
Example:

8. Click **Next**.
9. The New/Preview dialog box opens.

Example:

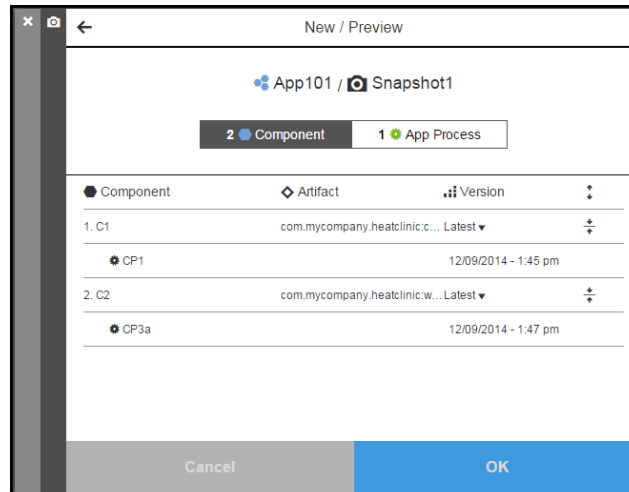
10. Toggle between the Component and App Process views to see the components and application processes for the selected application.

Example:



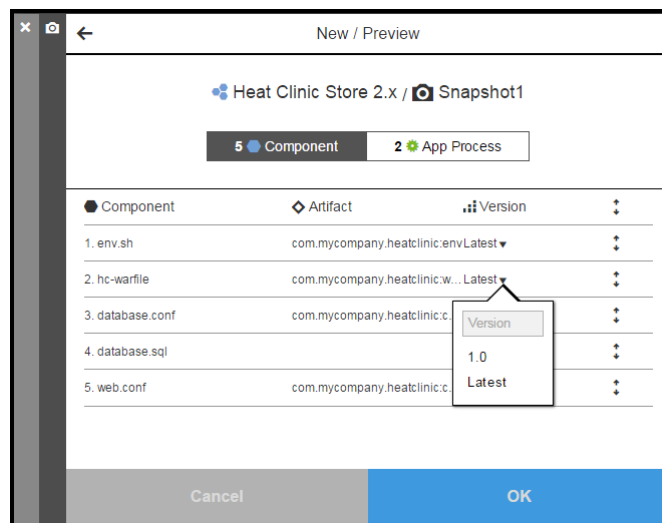
11. In the Component view, select the artifact versions for the snapshot:
 - a. Click the **View** button to see the component processes for a component.

Example:



- b. Click the down arrow in the **Version** column to open the drop-down list of available artifact versions.

Example:



- c. Select an artifact version.

- Click **OK**.

The "Snapshot <snapshot_name> has been created.>" message appears.

The Snapshot List is updated and now has the snapshot that you added.

Example:

Applications / App101		2 Snapshots		Delete	Compare	Select	All	Add
1	Snapshot1	Pre-alpha version	12/09/2014 - 2:09 PM	By admin				
2	Snapshot2	Automated test sample	12/09/2014 - 2:21 PM	By admin				

Deploying Snapshots

- Go to the Applications List.
- Choose an application.
- Click the **Run process** button.

Example:



- Select **New Run**, **Last Run**, or **Schedule**.

The dialog box to set the parameters for running an application opens.

In this dialog box, you can deploy a snapshot or compare the application to the selected snapshot.

Example:

Store 1.2

Select Process

Select Environment

5 Select a Snapshot

Compare

0 Artifacts

0 Selected

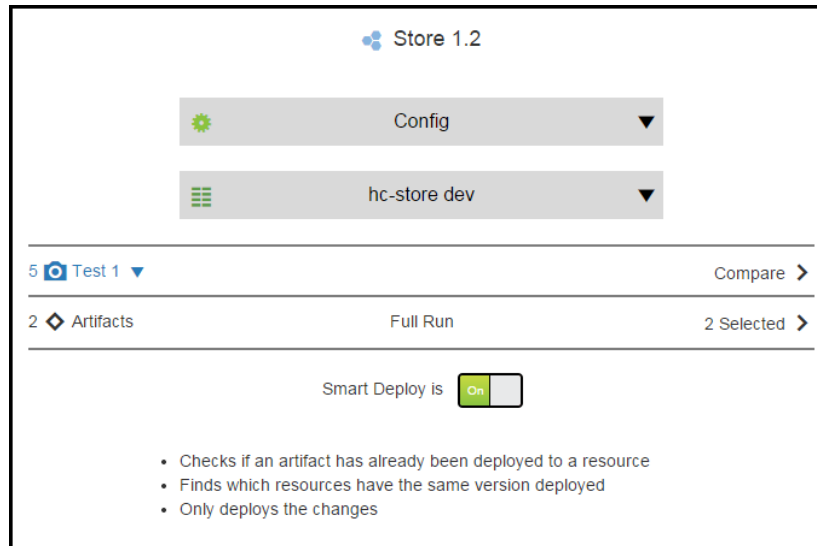
Smart Deploy is

Off

5. Select the parameters to run an application.

For more information, go to [Deploying or Comparing Snapshots](#) on page 508 and [Deploying \(Running\) Applications](#) on page 152.

Example:



6. To run (deploy) the snapshot:

1. Click **OK**.
2. Go to [Deploying \(Running\) Applications](#) on page 152 for more information.

7. To compare the application to the selected snapshots:

1. Click **Compare** to compare the application to the selected snapshot.
2. Go to [Comparing Snapshots](#) on page 511.

You can compare the application to other snapshots.

Deploying or Comparing Snapshots

How to get here: From the Home page, go to the Applications List, choose an application, click the Run process button, and select **New Run**.

Example:



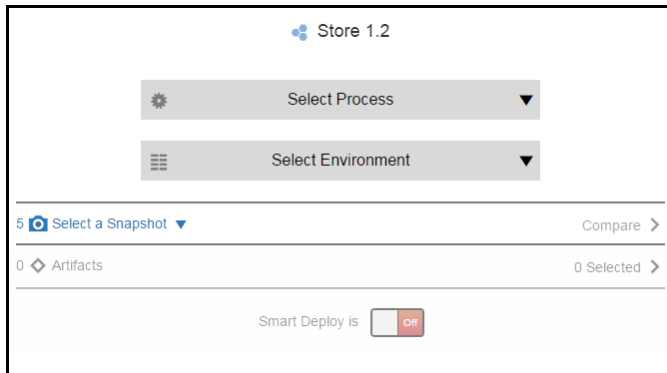
The dialog box to set the parameters for running (deploying) the application opens.

In this dialog box, you can deploy a snapshot or compare the application to a snapshot.

Setting Parameters in the Dialog Box

In the dialog box, the **Select a Snapshot** option is available (enabled) because the application has one or more snapshots saved.

Example:



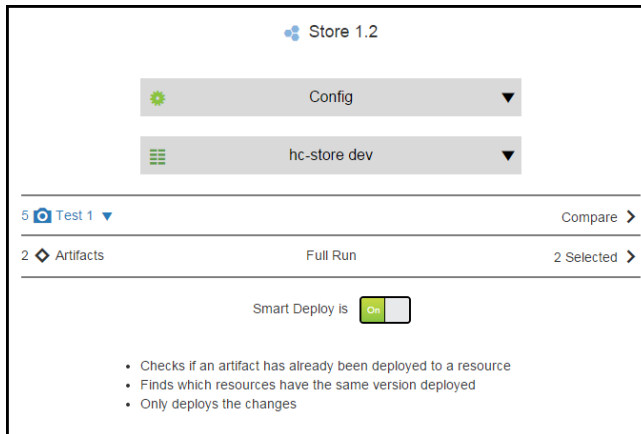
Select the following options to set the parameters to run the application:

- **Select Process**—Click the pull-down button to select the application to run.
- **Select Environment**—Click the pull-down button to select the environment in which the application will run.
- **Select a Snapshot**—Click the pull-down button to select one or more snapshot.

The **Compare** option is now available (enabled) because of the following

- There are one or more snapshots that can be compared to the application in the selected environment.
- The application has been selected.
- The environment has been selected.
- The one or more snapshot has been selected.

Example:



Deploying Snapshots

After setting the parameters, click **OK** to run (deploy) the selected snapshot.

For more information, go to

Comparing Snapshots

After setting the parameters, click **OK** to compare the selected application to a snapshot.

Managing Snapshots

You can do the following tasks on a Snapshot List:

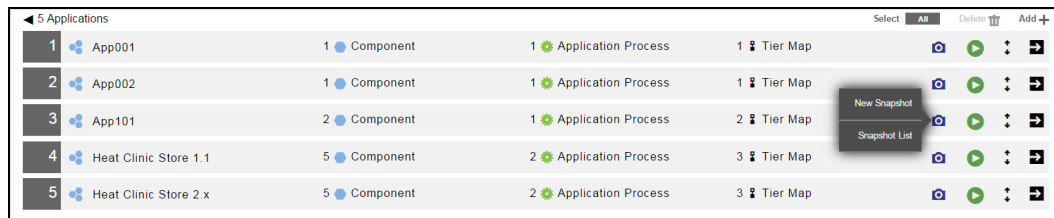
To open the Snapshot List:

1. Go to the Applications list.
2. Choose an application.
3. Click the **Snapshot** button.

Example:



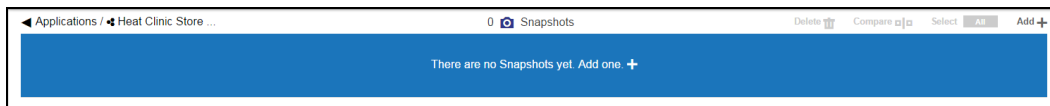
4. Select **Snapshot List**.



5 Applications				Select	All	Delete	Add +
1	App001	1 Component	1 Application Process	1 Tier Map			
2	App002	1 Component	1 Application Process	1 Tier Map			
3	App101	2 Component	1 Application Process	2 Tier Map			
4	Heat Clinic Store 1.1	5 Component	2 Application Process	3 Tier Map			
5	Heat Clinic Store 2.x	5 Component	2 Application Process	3 Tier Map			

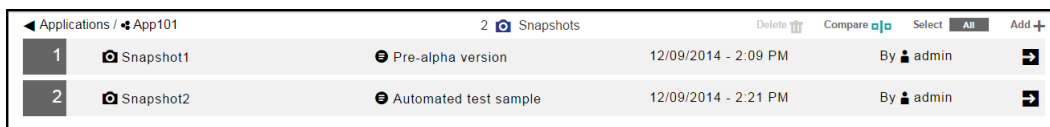
The Snapshot List opens.

If no snapshots have been saved, the Snapshot List is empty. Go to [Adding Snapshots](#) on page 503 to add a snapshot in the application.



Applications / Heat Clinic Store ...		0 Snapshots	Delete	Compare	Select	All	Add +
There are no Snapshots yet. Add one +							

If one or more snapshots have been saved, the Snapshot List shows the saved snapshots.



Applications / App101		2 Snapshots	Delete	Compare	Select	All	Add +
1	Snapshot1	Pre-alpha version	12/09/2014 - 2:09 PM	By admin			
2	Snapshot2	Automated test sample	12/09/2014 - 2:21 PM	By admin			

After you choose a snapshot, click the **View details** button to view more information about the snapshot.

Comparing Snapshots


Use one of these methods to compare snapshots:

- [Comparing an Application to Snapshots](#) on page 512
- [Comparing Snapshots](#) on page 514

Comparing an Application to Snapshots

1. Go to the Environments List.
2. Select an environment.

Example:

3	 hc-store dev	Enabled  	1  Applications installed	1 ...  
4	 hc-store prod	Enabled  	1  Applications installed	 
5	 hc-store qe	Enabled  	1  Applications installed	1 ...  

3. Click the **Inventory** button to open the Environment Inventory.

Example:



4. Choose an application.

Example:

Environments / Inventory		 hc-store dev	
1	 Store 1.2	3  Artifacts	1   

5. Click the **Snapshot** button.

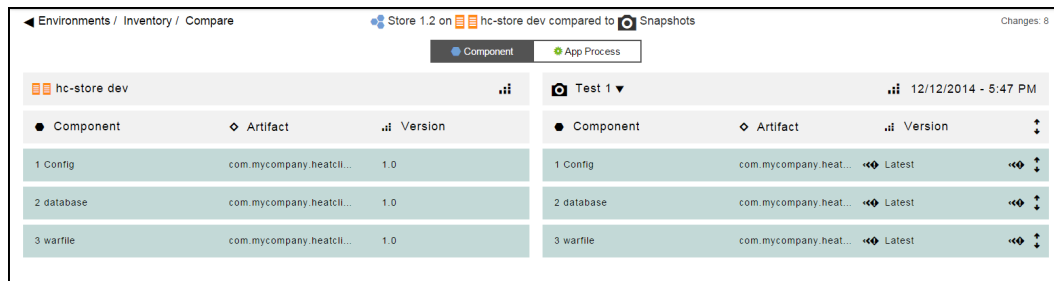
Example:



6. Select **Compare**.**Example:**

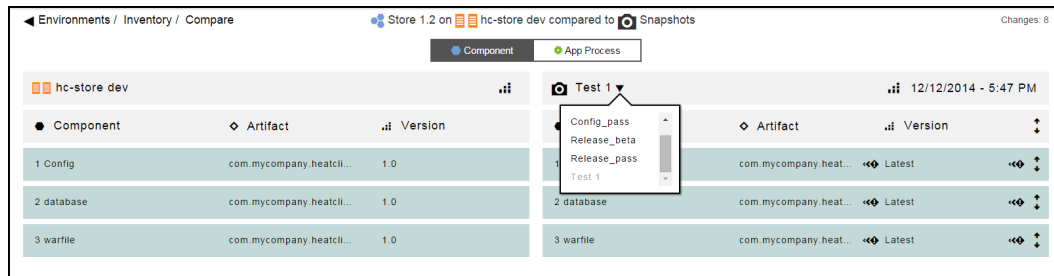
The Snapshot Comparison page opens. You can compare the currently deployed application to a snapshot.

The currently deployed application is on the left, and the snapshot is on the right. You can choose the snapshot for the comparison.

Example:

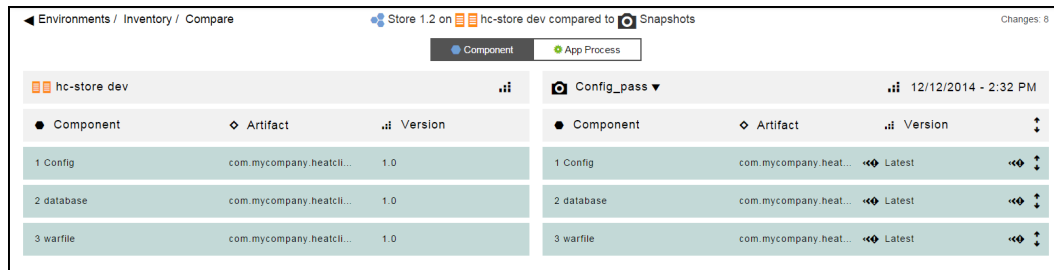
- Click the pull-down button next to the snapshot name to select a different snapshot.

Example:



The Snapshot Comparison page now shows the comparison between the application and the Config_pass snapshot. The components and component processes have changed.

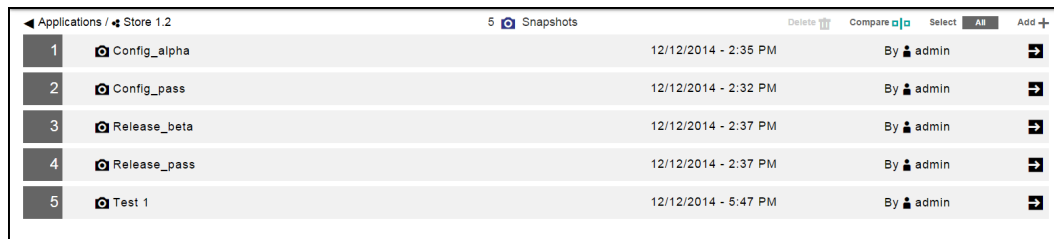
Example:



Comparing Snapshots

- Go to the Applications List.
- Select an application.
- Click the **Snapshot** icon.
- Select **Snapshot List**. The Snapshot Comparison page opens.

Example:



5. Click the **Compare Snapshots** button to open the Snapshot Comparison page.

Example:



Example:

Release_beta ▼ 12/12/2014 - 2:37 PM			Test 1 ▼ 12/12/2014 - 5:47 PM		
Component	Artifact	Version	Component	Artifact	Version
1 Config	com.mycompany.heatcli...	1.0	1 Config	com.mycompany.heatcli...	Latest
2 warfile	com.mycompany.heatcli...	1.0	2 warfile	com.mycompany.heatcli...	Latest
3 database	com.mycompany.heatcli...	Latest	3 database	com.mycompany.heatcli...	Latest

6. Click the pull-down button next to the snapshot name in a column to select other snapshots.

Example:

Release_beta ▼ 12/12/2014 - 2:37 PM			Test 1 ▼ 12/12/2014 - 5:47 PM		
Component	Artifact	Version	Component	Artifact	Version
1 Config	com.mycompany.heatcli...	1.0	1 Config_alpha	com.mycompany.heatcli...	Latest
2 warfile	com.mycompany.heatcli...	1.0	2 warfile	com.mycompany.heatcli...	Latest
3 database	com.mycompany.heatcli...	Latest	3 database	com.mycompany.heatcli...	Latest

The results change when you select different snapshots.

Example:

Release_beta ▼ 12/12/2014 - 2:37 PM			Config_alpha ▼ 12/12/2014 - 2:35 PM		
Component	Artifact	Version	Component	Artifact	Version
1 database	com.mycompany.heatcli...	Latest	1 database	com.mycompany.heatcli...	1.0
2 Config	com.mycompany.heatcli...	1.0	2 Config	com.mycompany.heatcli...	1.0
3 warfile	com.mycompany.heatcli...	1.0	3 warfile	com.mycompany.heatcli...	1.0

Comparing an Application to Snapshots

How to get here: From the Home page, go to the Environments List, select an environment, and click the **Inventory** button to open the Environment Inventory.

Example:



This example shows the Store 1.2 application.

5	Store 1.2	3	Component	2	Application Process	3	Tier Map				
8_Release	Release	hc-store prod	Dec 12, 2014	2:58 Pacif...	00:04	100%					
7_Config	Config	hc-store prod	Dec 12, 2014	2:57 Pacif...	00:18	100%					
6_Release	Release	hc-store dev	Dec 12, 2014	2:36 Pacif...	00:03	100%					
5_Config	Config	hc-store prod	Dec 12, 2014	2:34 Pacif...	00:18	100%					
4_Config	Config	hc-store prod	Dec 12, 2014	2:31 Pacif...	00:19	100%					
3_Release	Release	hc-store dev	Dec 12, 2014	2:30 Pacif...	00:05	100%					
2_Config	Config	hc-store qe	Dec 12, 2014	2:29 Pacif...	00:26	100%					
1_Config	Config	hc-store dev	Dec 12, 2014	2:28 Pacif...	00:20	100%					

The application was run in these environments:

3	hc-store dev	Enabled	1	Applications Installed	1		
4	hc-store prod	Enabled	1	Applications Installed			
5	hc-store qe	Enabled	1	Applications Installed	1		

The Environment Inventory for the environment called hc-store dev shows that three artifacts were deployed.

Environments / Inventory		hc-store dev			
1	Store 1.2	3	Artifacts	1	

After you click the **View** button, more information about the artifacts appear.

Environments / Inventory		hc-store dev			
1	Store 1.2	3	Artifacts	1	
database	com.my...config	1.0		41 minutes ago	1/1
Config	com.my...arfile	1.0		41 minutes ago	
warfile	com.my...arfile	1.0		33 minutes ago	

When you click the **Snapshot** button, you can either create a new snapshot or compare a snapshot to the currently deployed application.

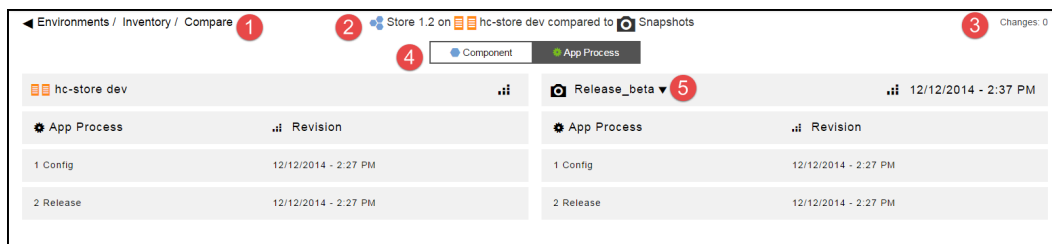
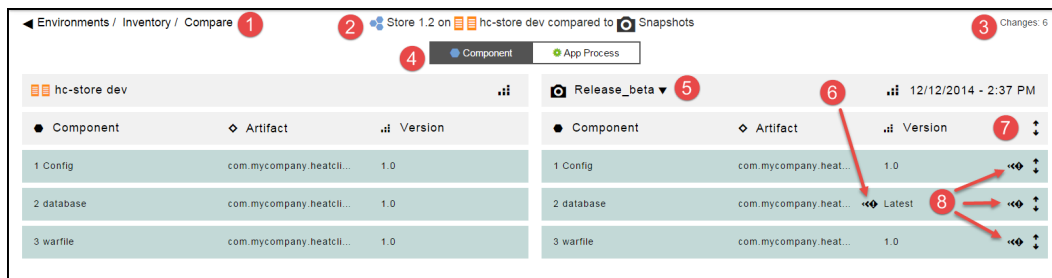
When one or more snapshots are saved for an application, this menu appears. If the application has no snapshots, you can only create a snapshot.

Example:



In the Snapshot Comparison page, you can compare the currently deployed application to a snapshot. The currently deployed application is on the left, and the snapshot is on the right. You can choose the snapshot for the comparison.

The Snapshot Comparison page has the following information:



1	Breadcrumb to the Snapshot Comparison page: <i>Environments / Inventory / Compare</i>
2	Page title: <Application name> on <Environment Name> compared to Snapshots
3	Number of changes, which is updated automatically depending on the selected snapshot.

4	Toggle between the Component and App Process views.
5	When more than one snapshot is saved, the pull-down button is available (enabled). You can select other snapshots to compare.
6	Change Alert icon, which appears next to the object that changed.
7	Click the View button to expand all the rows and view information about the objects in both columns.
8	Click the View button in each object row to view more information about the object in both columns.

When you expand the rows to show details about the objects, the comparison shows that the component processes changed between the currently deployment application and the Release_beta snapshot.

You can compare a different snapshot to the application. Click the pull-down button next to the snapshot name to select a snapshot.

The Snapshot Comparison page now shows the comparison between the application and the Config_pass snapshot. The components and component processes changed.

Environments / Inventory / Compare Store 1.2 on hc-store dev compared to Snapshots Changes: 8

Component App Process

Component	Artifact	Version
1 Config	com.mycompany.heatcli...	1.0
2 database	com.mycompany.heatcli...	1.0
3 warfile	com.mycompany.heatcli...	1.0
1 DBUpdate		12/12/2014 - 2:23 PM
2 Deploy file		12/12/2014 - 2:26 PM
3 Save		12/12/2014 - 2:30 PM

Component	Artifact	Version
1 Config	com.mycompany.heatcli...	Latest
1 Save		12/12/2014 - 2:26 PM
2 database	com.mycompany.heatcli...	Latest
1 DBUpdate		12/12/2014 - 2:30 PM
3 warfile	com.mycompany.heatcli...	Latest
Deleted		
2 Deploy file		12/12/2014 - 2:23 PM
Deleted		

Comparing Snapshots in the Snapshot List

How to get here: From the Applications List, select an application > click the **Snapshot** button > select **Snapshot List**.

Example:



This example shows the Store 1.2 application.

5	Store 1.2	3 Component	2 Application Process	3 Tier Map	
8_Release	Release	hc-store prod	Dec 12, 2014 2:58 Pacif...	00:04	100%
7_Config	Config	hc-store prod	Dec 12, 2014 2:57 Pacif...	00:18	100%
6_Release	Release	hc-store dev	Dec 12, 2014 2:36 Pacif...	00:03	100%
5_Config	Config	hc-store prod	Dec 12, 2014 2:34 Pacif...	00:18	100%
4_Config	Config	hc-store prod	Dec 12, 2014 2:31 Pacif...	00:19	100%
3_Release	Release	hc-store dev	Dec 12, 2014 2:30 Pacif...	00:05	100%
2_Config	Config	hc-store qe	Dec 12, 2014 2:29 Pacif...	00:26	100%
1_Config	Config	hc-store dev	Dec 12, 2014 2:28 Pacif...	00:20	100%

The Snapshot List has four snapshots:

Applications / Store 1.2		4 Snapshots	Delete	Compare	Select	All	Add
1	Config_alpha	12/12/2014 - 2:35 PM	By admin				
2	Config_pass	12/12/2014 - 2:32 PM	By admin				
3	Release_beta	12/12/2014 - 2:37 PM	By admin				
4	Release_pass	12/12/2014 - 2:37 PM	By admin				

Click the **Compare Snapshots** button to open the Snapshot Comparison Page.

Example:



In this Snapshot Comparison page, you can compare two snapshots. The newest snapshot is on the left, and the snapshot that was saved before the previous one is on the right.

The Snapshot Comparison page has the following information:

Applications / Store 1.2 / Snapshots compare

Store 1.2

Changes: 2

Component

App Process

Release_pass

12/12/2014 - 2:37 PM

Release_beta

12/12/2014 - 2:37 PM

Component	Artifact	Version
1 Config	com.mycompany.heatcli...	Latest
2 warfile	com.mycompany.heatcli...	Latest
3 database	com.mycompany.heatcli...	Latest

Component	Artifact	Version
1 Config	com.mycompany.heatcli...	1.0
2 warfile	com.mycompany.heatcli...	1.0
3 database	com.mycompany.heatcli...	Latest

Applications / Store 1.2 / Snapshots compare

Store 1.2

Changes: 0

Component

App Process

Release_pass

12/12/2014 - 2:37 PM

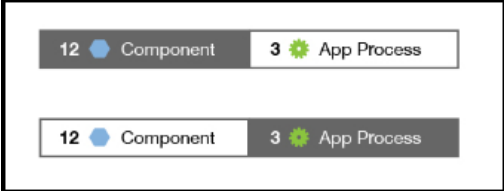

Release_beta

12/12/2014 - 2:37 PM

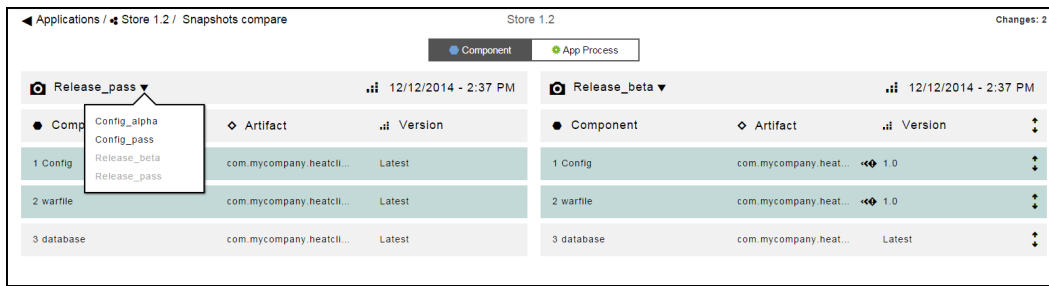
App Process	Revision
1 Config	12/12/2014 - 2:27 PM
2 Release	12/12/2014 - 2:27 PM

App Process	Revision
1 Config	12/12/2014 - 2:27 PM
2 Release	12/12/2014 - 2:27 PM

1	Breadcrumb to this page: <i>Applications / <Application name>/ Snapshots compare</i>
2	Application name
3	Number of changes, which is updated automatically depending on the selected snapshots.

4	<p>Toggle between the Component, App Process, and Environment views.</p> <p>Before a snapshot is deployed, only the Component and App Process views are available.</p>  <p>After the snapshot is deployed, Component, App Process, and Environment views are available.</p> 
5	<p>When more than two snapshots are saved, the pull-down button is available (enabled). You can select other snapshots to compare.</p>
6	<p>When more than two snapshots are saved, the pull-down button is available (enabled). You can select other snapshots to compare.</p>
7	<p>Change Alert icon, which appears next to the object that changed.</p>
8	<p>Click the View button to expand all the rows and view information about the objects in both columns.</p>
9	<p>Click the View button in each object row to view more information about the object in both columns.</p>

You can compare other snapshots to compare. Click the pull-down button next to the snapshot name in a column to select other snapshots.



Applications / Store 1.2 / Snapshots compare

Store 1.2

Changes: 2

Component App Process

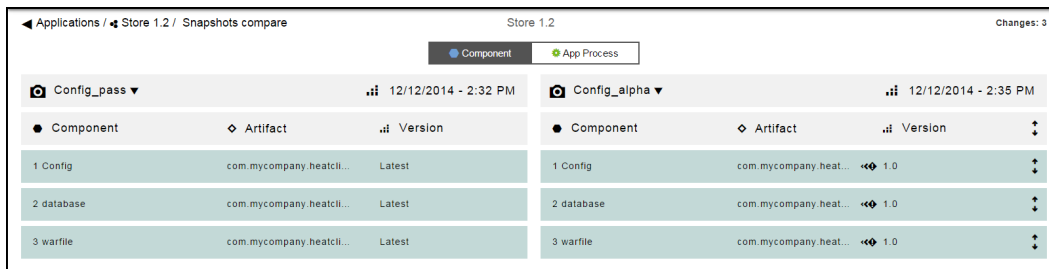
Release_pass 12/12/2014 - 2:37 PM

Release_beta 12/12/2014 - 2:37 PM

Component	Artifact	Version
1 Config	com.mycompany.heatcli...	Latest
2 warfile	com.mycompany.heatcli...	Latest
3 database	com.mycompany.heatcli...	Latest

Component	Artifact	Version
1 Config	com.mycompany.heatcli...	1.0
2 warfile	com.mycompany.heatcli...	1.0
3 database	com.mycompany.heatcli...	Latest

The results change when you select different snapshots:



Applications / Store 1.2 / Snapshots compare

Store 1.2

Changes: 3

Component App Process

Config_pass 12/12/2014 - 2:32 PM

Config_alpha 12/12/2014 - 2:35 PM

Component	Artifact	Version
1 Config	com.mycompany.heatcli...	Latest
2 database	com.mycompany.heatcli...	Latest
3 warfile	com.mycompany.heatcli...	Latest

Component	Artifact	Version
1 Config	com.mycompany.heatcli...	1.0
2 database	com.mycompany.heatcli...	1.0
3 warfile	com.mycompany.heatcli...	1.0

Deleting Snapshots

Do one of these methods to delete snapshots:

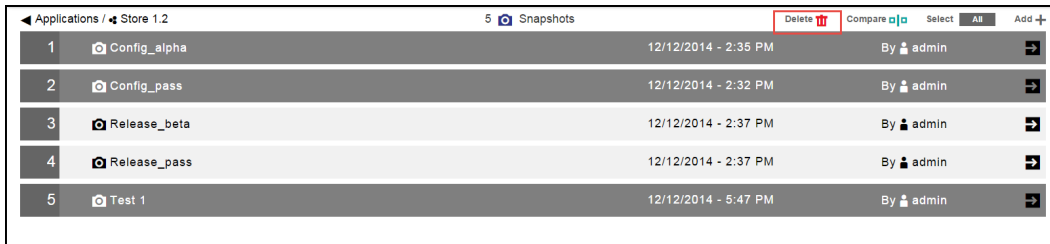
- [Selecting Snapshots to Delete](#) on page 522
- [Deleting All the Snapshots](#) on page 523

Selecting Snapshots to Delete

To delete one or more snapshots:

1. Select the snapshots that you want to delete.
2. Click the Delete button.

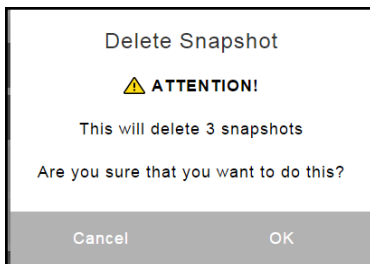
Example:



The screenshot shows a table titled 'Snapshots' with 5 items. The 'Delete' button in the top right corner is highlighted with a red box.

1	Config_alpha	12/12/2014 - 2:35 PM	By admin	
2	Config_pass	12/12/2014 - 2:32 PM	By admin	
3	Release_beta	12/12/2014 - 2:37 PM	By admin	
4	Release_pass	12/12/2014 - 2:37 PM	By admin	
5	Test 1	12/12/2014 - 5:47 PM	By admin	

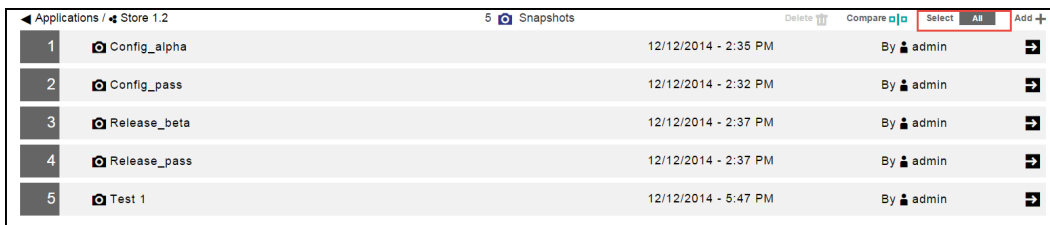
A message appears.



3. Click **OK**.

Deleting All the Snapshots

1. Click **All**.

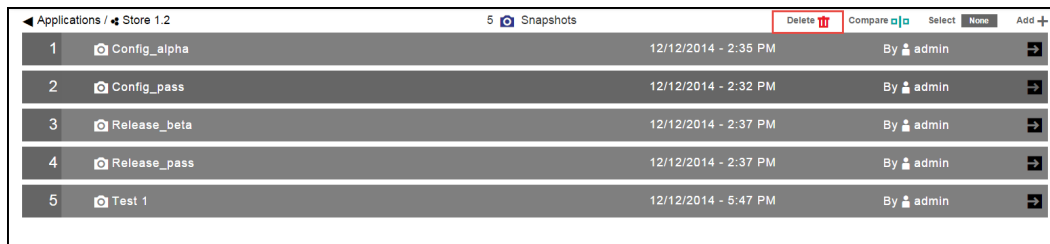


The screenshot shows the same 'Snapshots' table as before. The 'All' button in the top right corner is highlighted with a red box.

1	Config_alpha	12/12/2014 - 2:35 PM	By admin	
2	Config_pass	12/12/2014 - 2:32 PM	By admin	
3	Release_beta	12/12/2014 - 2:37 PM	By admin	
4	Release_pass	12/12/2014 - 2:37 PM	By admin	
5	Test 1	12/12/2014 - 5:47 PM	By admin	

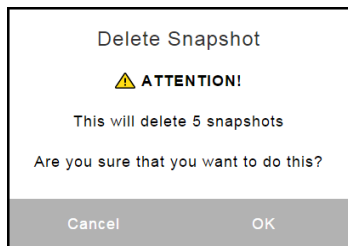
All of the snapshots are selected.

2. Click the Delete button.



Applications / Store 1.2		5 Snapshots	Delete	Compare	Select	Now	Add
1	Config_alpha	12/12/2014 - 2:35 PM	By admin				
2	Config_pass	12/12/2014 - 2:32 PM	By admin				
3	Release_beta	12/12/2014 - 2:37 PM	By admin				
4	Release_pass	12/12/2014 - 2:37 PM	By admin				
5	Test 1	12/12/2014 - 5:47 PM	By admin				

A message appears.



3. Click **OK**.

Automation Platform Tasks

To configure ElectricFlow to perform tasks such as:

- Tracking the jobs.
- Retrieving information about job steps.
- Viewing jobs results.
- Creating and managing projects and workspaces.
- Defining procedures that contain steps and subprocedures.
- Determining when to execute procedures based on a schedule or event.
- Managing artifacts and artifact versions.
- Managing plugins.

Go to the [Automation Platform Basics](#) on page 526.

Pipeline Automation

Coming soon ...

Automation Platform Basics

ElectricFlow is built on a powerful proven automation platform that natively integrates domain-specific capabilities for Enterprise-level continuous delivery. The automation platform gives distributed DevOps teams shared control and visibility into infrastructure, tool chains, and processes. It accelerates and automates the software delivery process and enables agility, availability, predictability, and security across many build-test, deployment, and release pipelines.

This section describes how to create, configure, and manage objects in the automation platform that make the automation of build-test processes, application deployments, and pipelines possible in ElectricFlow. Examples include:

- For build-test automation: Configuring resources and creating procedures to build and test your software.
- For deployment automation: Creating components based on artifacts, defining process steps with plugins, and assigning resources to environments.
- For pipeline automation: Defining pipeline tasks based on procedures.

For more information about the ElectricFlow objects, concepts, and features in this topic, go to the [ElectricFlow Glossary](#) on page 53

You use only the automation platform UI to configure ElectricFlow to perform these tasks. The objects and operations for build-test automation are the same as those for the automation platform.

You can also use Perl API commands through `ec-perl` and `ectool`, RESTful API commands, and DSL scripts. For information about using API commands and DSL scripts, see the [ElectricFlow API Guide](#).

What the Automation Platform Does

Before you use ElectricFlow for build-test automation, deployment automation, or pipeline automation, you must create, configure, and manage these objects in the automation platform:

For build-test automation, you must create, configure, and manage these objects in the automation platform:

- Projects

A *project* is an object used in ElectricFlow to organize information. A project is a container object for procedures, steps, schedules, workflows, and properties. If you use ElectricFlow for different purposes, you can use a separate project for each purpose so different projects do not interfere with each other. When you work in one project, you do not normally see information in other projects. At the same time, a project can use information defined in other projects, which allows you to create shared library projects.

- Resources

A *resource* is a server machine available to ElectricFlow for running steps. A resource has a logical name and a host name. In some situations, it is convenient to have multiple logical resources associated with the same host. A resource can also be associated with one or more pools. Each resource has a *step limit* that determines the maximum number of steps that can execute simultaneously on the resource. Resources can be grouped into *resource pools*.

- Procedures

Procedures and *steps* define tasks that you want ElectricFlow to execute. A procedure consists of one or more steps. A step includes a command or script executed on a single resource and is the smallest unit of work that ElectricFlow understands. Each step specifies a resource on which it should run (either the name of a specific machine or the name of a *resource pool* of equivalent machines, in which case ElectricFlow picks a machine from the pool). A step can be given a time limit, and if the step does not complete within the specified time, ElectricFlow automatically aborts it.

Steps are ordered within a procedure and normally execute sequentially. However, it is possible to mark a consecutive range of steps for parallel execution, so all steps in that range run concurrently.

You can define *parameters* for procedures. Parameter values are assigned when procedures are scheduled. Parameters can be required, optional, or have default values. Parameters are used for a variety of purposes such as specifying the branch to build or the set of platforms on which to run tests. Parameter values can be used in step commands and many other places.

Procedures can be nested. A step in one procedure can invoke another procedure instead of running a command. The invoking step provides parameters needed by the nested procedure, also referred to as a subprocedure.

- Schedules

A schedule is used to execute procedures and determine when specific procedures run. A schedule can trigger at defined times, for example, every 2 hours from 10:00 pm to 6:00 am on Mondays, Wednesdays, and Fridays, or when modifications are checked into a particular branch of your source code control system. It is also possible to create a schedule that runs immediately and disappears after the job runs. When you create a schedule, you must provide the parameters required by the procedure that you want to invoke.

The Continuous Integration Dashboard works with your source code management (SCM) system and provides visibility into running builds, the ability to add a project to continuous integration quickly, and easily accessed configuration pages to setup or modify a continuous integration schedule.

- Workflows

Managing a build-test-deploy product life cycle spanning multiple procedures and projects requires a significant amount of "meta-programming" and a heavy use of properties, and the *workflow* feature simplifies this process. Using the workflow object, you can create build-test-deploy life cycles by defining a set of states and transitions. Any ElectricFlow project can contain a workflow.

- When a procedure is executed or run, a *job* is created. A job is an object that is created each time a procedure begins to execute or run. The job keeps track of all data associated with the procedure's execution, such as the running time of each step and any errors that may occur during the step. ElectricFlow retains job information after the job completes so you can examine what occurred.

After setting resources, procedures, and schedules, ElectricFlow automatically runs the procedures that you created using these objects and facilities:

- Zones and Gateways—A zone (or top-level network) that you create is a way to partition a collection of agents to secure them from use by other groups. A gateway is a secured connection between two zones when you want to share or transfer information between the zones. For example, you might want a developers zone and a test zone. The ElectricFlow server is a member of the *default* zone, created during ElectricFlow installation.

- *Continuous Integration Builds* and other schedules—Run jobs according to *schedules* that you define. Scheduled jobs can run at specific times or when source code changes are checked in to your source control system. ElectricFlow integrates with major source control systems. The Continuous Integration Dashboard allows you to add more projects easily and create build configurations quickly so you can visually see running builds, build status, and so on.
- *Artifact Management* functionality—Using artifacts can improve performance across builds, provide better reusability of components, and improve cross-team collaboration with greater tractability. For example, instead of developers repeatedly downloading third-party packages from external sources, these components can be published and versioned as an artifact. Developers then simply retrieve a specific artifact version from a local repository, guaranteeing a consistent package from build to build.
- *Preflight build functionality*—Used by developers to build and test code changes in isolation on their local machines before those changes are committed to a production build.
- *Plugin* capability—ElectricFlow is built with an extensible UI, enabling easy development of plugins that include integrations with other tools, custom dashboards, and unique user experiences based on roles. "Bundled" plugins, installed during ElectricFlow installation, provide easy integration with your SCM systems, defect tracking applications, and so on.
- *Workflow* functionality—Use a workflow to design and manage processes at a higher level than individual jobs. You can use workflows to combine procedures into processes to create build-test-deploy life cycles (for example). A workflow contains states and transitions that you define to provide complete control over your workflow process. The ElectricFlow Workflow feature allows you to define an unlimited range of large or small life cycle combinations to meet your needs.
- *Resource* management—If a resource is overcommitted, ElectricFlow delays some jobs until others are finished with the resource. You can define pools of equivalent resources, and ElectricFlow spreads usage across the pool.
- Recording a variety of information about each job, such as the running time and the success or failure of each step. A set of reports is available to provide even more information.
- Powerful and flexible *reporting* facilities—Various statistics such as number of compiles or test errors are collected after each step and recorded in the ElectricFlow database. A variety of reports can be generated from this information.
- Allowing you to observe jobs as they run and to cancel jobs or change their priorities.
- *Workspace* for each job, which is a disk area a job uses for storage—ElectricFlow also provides a facility for reclaiming space occupied by workspaces.
- Powerful data model based on *properties*—Properties are used to store job input data such as the source code branch to use for the build, to collect data during a job (such as number of errors or warnings), and to annotate the job after it completes (for example, a build has passed QA).
- *Access control* for users logged into the system—ElectricFlow uses this information to control their activities and integrates with Active Directory and LDAP repositories.
- *Search, sort, and filter* functions to minimize viewing or "wading" through information that is of no interest to you, allowing you to access the information you need quickly.
- *Email notifications* to get important information or data to individuals or groups immediately and on a regular basis for a particular job or a specific job aspect.
- All ElectricFlow operations and features are available from a command-line application tool (Perl API), *ectool*, the RESTful API, DSL methods, and a user interface (UI).

For more information, go to [ElectricFlow Platform Objects and Functionality](#) on page 908.

Getting Started

[Overview](#)

[Basic Automation Platform Terminology](#)

[Getting Started Scenarios Help Topic Series](#)

[Navigating the ElectricFlow User Interface](#)

Overview

ElectricFlow is an application for automating and managing your development life cycle, the software build, test, and deployment process. It helps development teams make these tasks repeatable, visible, and efficient.

ElectricFlow creates an environment where IT and Development organizations can work together to connect physical and virtual environments, processes, and tools already in use to create a private development cloud, also known as a *smart development cloud*. ElectricFlow is a scalable solution, solving some of the biggest challenges of managing "back-end" software development tasks.

Basic Automation Platform Terminology

To get an overview of how the automation platform works and understand the ElectricFlow concepts and processes, review the following basic terms:

- [Project](#)
- [Procedure](#)
- [Step](#)
- [Plugins](#)
- [Parameter](#)
- [Schedule](#)
- [Continuous Integration](#)
- [Resource](#)
- [Job](#)
- [Report](#)
- [Workspace](#)
- [Zones](#)
- [Gateways](#)

See the [glossary](#) for a comprehensive list of ElectricFlow terminology.

Getting Started Scenarios Help Topic Series

The series of scenario Help topics are designed to help you learn to use the ElectricFlow automation platform quickly. Follow the step-by-step scenario format to learn the automation platform basics:

- Creating a simple procedure—[Scenario 1](#)
- Creating a procedure that uses a Source Code Management (SCM) system— [Scenario 2](#)
- Setting up email notifications, reporting, and scheduling—[Scenario 3](#)
- Creating a multi-agent build procedure—[Scenario 4](#)

Each scenario builds on what you learned in the previous scenario, and each scenario ends with one or more scenario extensions to introduce you to other related ElectricFlow functionality.

Navigating the Automation Platform User Interface

To quickly link to task pages, view running or completed jobs, configure resources and so on, the automation platform web interface displays tabs across the top of the page. These tabs remain available at the top of all ElectricFlow automation-platform web pages.

When you select some of the main tabs, subtabs are provided. For example (see the next screen example), when the **Home** tab is selected, notice that the Overview tab is in bold font, which means you are looking at the Overview page. This page can be customized to show the information that you want to see quickly. You can use this page for shortcuts, quick links to jobs, "thumbnail" report views, and so on.

Selecting the Continuous Integration subtab takes you to the Continuous Integration Dashboard, which is place to configure your projects for continuous integration builds.

In addition to the main set of tabs, the top bar includes:

- **Logged in as** clearly shows the name of the logged-in user.
- **Logout**—Use this link to log out between ElectricFlow sessions if necessary.
- **Help**—This link provides a Help topic for the current page you are viewing, but if you select the down-arrow adjacent to the Help link, you have access to the following:
 - **Tutorials**—Use this link to display the Table of Contents for all currently available tutorial examples you may find useful as you become familiar with ElectricFlow .
 - **Documentation**—Use this link to access the entire ElectricFlow online Help system, which is fully searchable in the event you do not quickly see what you are looking for in the left-pane Table of Contents.

The Help Table of Contents contains feature overview/concept user-guide style Help topics, which are not linked to a particular web page. These topics are listed in the Table of Contents *above* the "Web Interface Help" section. The Web Interface Help section contains all of the page-specific Help topics.

Note: *To print a Help topic*— Right-click your mouse in the Help topic pane or go to your browser File menu.

- **Visit the Community Site**—This link takes you to the Electric Cloud Knowledge Base, all product documentation, and so on.

- **Contact Support**—Use this link to access the Electric Cloud Technical support phone number and email address. At a later time, you may choose to reconfigure this link to redirect it to your internal support time. See the "Reconfigure Contact Support" section in the "[Customizing the ElectricFlow UI](#)" Help topic.
- **About**—Use this link to display the current version of ElectricFlow you are using.

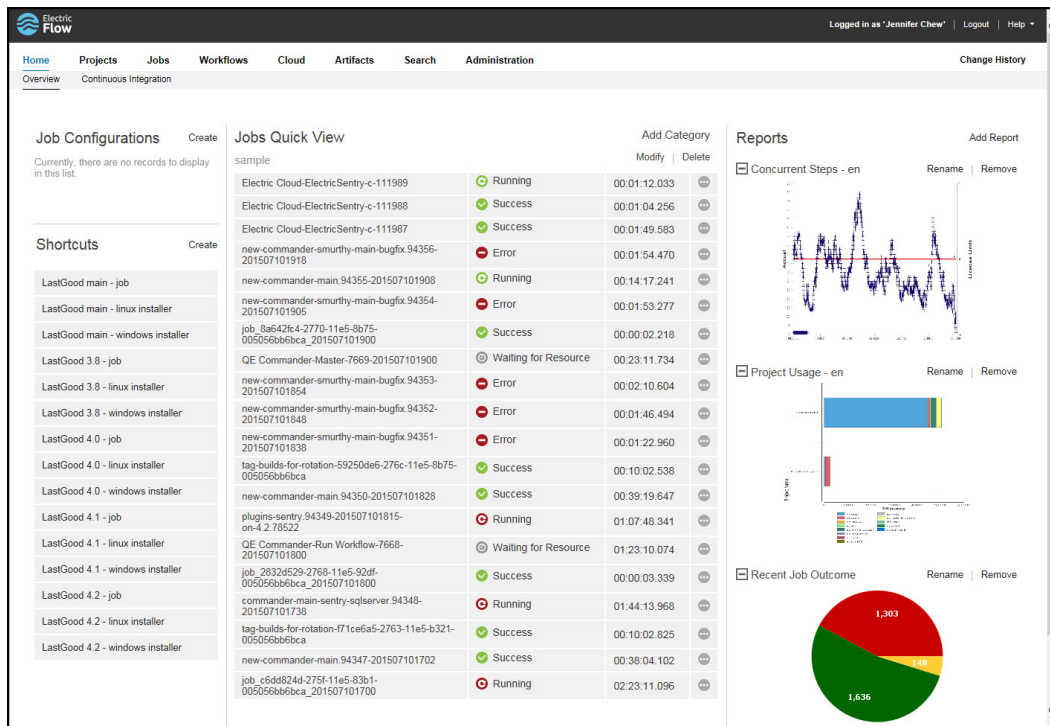
Home Tab

You can customize the Home page to see the information you need at-a-glance or add Shortcuts to quickly return to the ElectricFlow pages you visit frequently. The Home page provides four categories:

- **Job Configurations**
ElectricFlow procedures can contain complex sets of parameters, making it difficult to remember the correct parameters for a particular situation and tedious to re-enter those parameters every time the procedure is invoked. Job Configurations provide a one-click solution to this problem.
- **Shortcuts**
Use shortcuts to save frequently visited ElectricFlow web pages, so those pages are immediately accessible. You can also create a shortcut to any page on the web.
- **Jobs Quick View**
The Jobs Quick View allows you to define job categories that are interesting to you. You may be interested in a few jobs on this server. For example, you want only information about jobs that you launch manually and official builds for the products you work on. You do not want information about production builds for other products or personal jobs for other users.
- **Reports**
You can configure reports you would like to see on a regular basis and display a thumbnail report graphic in this section.

In the following Home page example, there are additional links. This page allows you to create, edit, and delete objects you see so that you can easily keep the page updated and current.

For more information about Home page functionality, see the [Home Page](#) Help topic.



Continuous Integration Subtab

The Continuous Integration Manager (CI Manager) provides a front-end user interface (the Continuous Integration Dashboard) for creating, managing, and monitoring continuous integration builds.

The Continuous Integration Dashboard provides:

- Visually seeing your running builds, build progress, build status, and accumulated build status.
- Easily accessed "Actions" to configure a continuous integration build.
- Quick configuration of your preferred SCM system.
- A project that contains any number of continuous integration builds, depending on the work you have already setup for your procedures and steps to perform.

The following is an example of the dashboard. Your initial view will be blank until you add configurations (using your preferred SCM) to run your build procedures continuously. When this tab is opened to view the dashboard, click the **Help** link in the upper-right corner of the page for instructions to begin using the dashboard.

Continuous Integration Dashboard

Filter by: ☐ Broken ☒ Running Update Filter [Settings](#) [Add Project](#)

Projects/Configurations	Recent Success Ratio	Jobs (Oldest -> Newest)	Current Job Progress	Actions
[-] BundledPlugins				[Menu Icon]
nightly-full application123Main	● 0%			[Menu Icon]
nightly-full-4.2	● 0%			[Menu Icon]
nightly-full-5.0	● 0%			[Menu Icon]

Projects, Jobs, and Workflow Tabs

Selecting the Projects, Jobs, or Workflows tab displays a table listing all previously configured projects, all completed jobs, or all previously configured workflows, respectively. Each table contains links in the table and links in the section above the table.

For example, selecting the Projects tab displays a table (see the next screen example), which has the projects you created and ElectricFlow default projects, which were added during ElectricFlow installation.

- Links at the top of the table include:
 - A "star" icon—Click this icon to add this page to your Home page.
 - A drop-down menu to choose which projects you want to see.
 - Create Project link to go to the New Project page to define and add a new project.
 - New Search link to find a project whose name you may not remember or projects of the same type.
- Links within the table include:
 - Click on a Project name (first column) to go to the Projects Details page for that project.
 - Actions column—This column usually contains links to Track Changes, Edit, Copy, or Delete the object, which is a project in this example.

Each table, Projects, Jobs, or Workflows, has similar links to help you get more information, modify existing information, or create a new object. For more information on any of these ElectricFlow pages, see these Help topics: [Projects](#), [Jobs](#), or [Workflows](#).

Projects ★				
115 Results				
			All Projects ▼	Create Project New Search
Project	Description	Impersonation Credential	Create Date	Actions
AM_Test			2015-07-10 17:54:12 PDT	
AWS-EC2	Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. We have a plugin, EC-EC2, that connects to it. This is the friendly front end, complete with a Workflow.		2012-11-14 13:23:29 PST	
AccuRev			2011-05-19 10:15:35 PDT	
Agent Tests	Tests to run on an agent before adding it to the ec-win pool.		2007-03-07 20:11:59 PST	

Cloud Tab

This tab opens to the Resources page and provides several other subtabs—Pools, Workspaces, Zones, Gateways, and the Cloud Manager plugin if it is installed.

Resources Page

The following example displays all resources that you configured ElectricFlow to use.

- In the information immediately after the **Resources** title at the top of the table, you can see how many licensed resources are in use.
- Use the row of buttons at the top of the page to perform resource management tasks.
- A resource name, in the **Name** column in the table, is a link to open the **Resource Details** panel for more information about the resource, and to access the **Edit Resource** panel to change or add access control privileges.

See the [Resources](#) Help topic for more information.

Resources / Current License Usage: 6 of 100 Concurrent Hosts; 0 of 100 Proxied Hosts

Filters

Resource Filters

Save Filters

Reset

Quick Search

► Status

Both Enabled/Disabled

Pools

Hosts

Step Limit

Proxy Agent

Filter

		Name	Pools	Type	Resource Template Name	Time running	Job Status	Description	Zone	HTTPS & Host	Step Load	Actions
		+++		Static			This agent has not been pinged yet	project: Flow-GE-ART branch: main purpose: QE Test Engineering	default		0 of unlimited	
		art-1	flow_ge_pool	Static				External ip 192.168.6.100 In order to connect to this vm use address 192.168.7.172	default	192.168.6.100	0 of unlimited	
		art-2	flow_ge_pool	Static				project: Flow-GE-ART branch: main purpose: QE Test Engineering	default	192.168.6.246	0 of unlimited	

Pools Subtab

Click this subtab to display a list of all resource pools available to ElectricFlow. You may find it useful to group resources, creating one or more resource pools. For example, you can have resource pools for specific purposes, development teams, or other groups in your organization.

Similar to the Resources page, this page provides:

- A link to create a new resource pool.
- A Search link to find an existing resource pool.
- Summary information for each resource pool in the list.
- Clicking the resource Pool Name takes you to the Resource Pool Details page for more information about that pool.
- And the Action column contains Copy and Delete links.

For more information about pools, see the [Resource Pools](#) or [Resource Pool - create new or edit existing pool](#) Help topics.

Workspaces Subtab

Similar to the Resources and Pools pages, the Workspaces page displays a table all available workspaces and summary information for each workspace.

Available links are:

- **Create Workspace** for adding another workspace.
- **New Search** to find an existing workspace.
- Clicking the Workspace Name takes you to the Edit Workspace page to make modifications.
- And the Action column contains **Copy** and **Delete** links.

For more information, see the [Workspaces and Disk Space Management](#), [Workspaces](#), or [Workspaces - create new or edit existing workspace](#) Help topics.

Zones and Gateways Subtabs

Similar to the other subtabs under the Cloud tab, the Zones and Gateways subtabs each have a table, listing all available zone or gateways, respectively.

- Selecting an object in the Name column takes you to the "Detail" page for more information.
- These pages are your management centers for zones or gateways, respectively.

For more information, see the [Zones](#) and [Gateways](#) Help topics.

Artifacts and Search Tabs

This page displays all artifacts available on this ElectricFlow server. It has Projects, Jobs, and Workflows pages, Create, Edit, and Delete links for the artifact. Clicking on an artifact Name (first column) takes you to the Artifact Details page. For more information, see the [Artifact Management](#) Help topic.

Home	Projects	Jobs	Workflows	Cloud	Artifacts	Search	Administration	Change History
Artifacts	Artifact Versions	Repositories						
Artifacts								
360 Results								
New Search Create Artifact								
Name	Group Id	Artifact Key	Description	Actions				
EC-Tutorials:MyArtifact	EC-Tutorials	MyArtifact						
QE:MyArtifact	QE	MyArtifact						
com.electriccloud.clusterRelatedFiles:clusterRelatedFiles	com.electriccloud.clusterRelatedFiles	clusterRelatedFiles						
com.electriccloud:EC-ALM	com.electriccloud	EC-ALM						

Artifact Versions Subtab

The Artifact Versions page displays all artifact versions available on the ElectricFlow server. Links in the table are the similar to other ElectricFlow pages. For more information, see the [Artifact Versions](#) Help topic.

Repositories Subtab

The Repositories page displays all artifact repositories available to the ElectricFlow server. For more information about repositories, see the [Artifact Management](#) and [Repositories](#) Help topics.

Search Tab

While many ElectricFlow web pages have a Search link, some do not. The Search page is always available. You can search any ElectricFlow supported object type on this page. For more information, see the [Define Search](#) Help topic.

Administration Tab

The next screen example shows all the current subtabs available when you select the Administration tab. Less-frequently used tasks are grouped for you as subtabs under the Administration tab. For example, after users and groups are set up, you do not need these pages unless a user or group configuration changes.

Home	Projects	Jobs	Workflows	Cloud	Artifacts	Search	Administration									Change History
Event Log	Groups	Users	Licenses	Directory Providers	Email Configurations	Database Configuration	Plugins	Server	Defect Tracking	Source Control						

Administration subtab descriptions:

- **Event Log**—This page displays a log of events generated anywhere in the system, including jobs and workflows.
- **Groups**—Use this page to view a filtered list of ElectricFlow *local* users or groups.
- **Users**—Use this page to view a filtered list of ElectricFlow *local* users or groups.
- **Licenses**—This page displays all license information known to the ElectricFlow server. Typically, a single license is displayed, which describes the usage to which you are entitled. The license on the server may be based on concurrent resources, concurrent users, concurrent steps, registered hosts, or any combination of these licenses.
- **Directory Providers**—ElectricFlow uses account information from multiple sources. In most cases, the primary account information source is an external LDAP or Active Directory repository: both user and group information is retrieved from the repository. *Local* users and groups are defined in ElectricFlow.
- **Email Configurations**—This page displays all previously configured email configurations. You must create an email configuration so ElectricFlow can communicate with your mail server to send email notifications
- **Database Configuration**—If you do not use the ElectricFlow-supplied default database, use this page to configure another database to communicate with ElectricFlow. See the *ElectricFlow Installation Guide* for a list of approved databases for use with ElectricFlow.
- **Plugins**—This subtab displays the Plugin Manager page. A *plugin* is a collection of one or more features that can be added to ElectricFlow. Numerous plugins are bundled and installed with ElectricFlow, integrating third-party products seamlessly with ElectricFlow. For example, Source Code Management plugins are installed and so that you can configure your preferred SCM to communicate with ElectricFlow.

- **Server**—This page displays overall information about the ElectricFlow server. Use the Settings or Access Control links to make changes.
- **Source Control**—This page displays all SCM (source code management) configurations you have created to communicate with ElectricFlow.
- **Defect Tracking** - This page displays previously configured Defect Tracking systems known to the ElectricFlow server. Use plugins to integrate ElectricFlow with numerous defect tracking systems.

Go to [Scenario 1](#).

Getting Started - Scenario 1 - Creating a Simple Procedure

To begin, select the Projects tab, then the **Create Project** link at the top of the table. You need to create a project to contain the procedures you create.

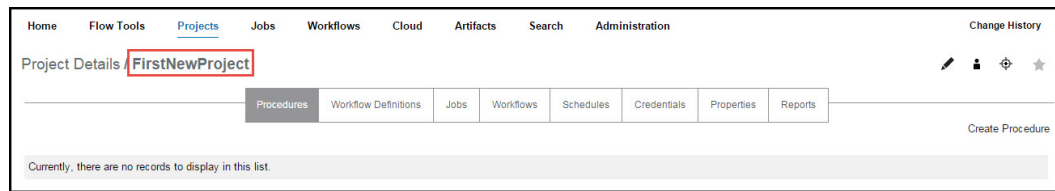
On the New Project page, enter information in the fields as follows:

Field Name	Description
Name	Type a unique project name. You may want your project names to reflect the work groups or teams that will be using them. For example, you might set project names based on the products they support. For our scenario examples, we will use "FirstNewProject" for our project name.
Description	Enter a text description for your reference if you choose. ElectricFlow does not use or interpret this information.
Default Resource	(optional) For the purpose of "getting started", leave this field blank to use the Default resource already created during the ElectricFlow installation.
Default Workspace	(optional) For the purpose of "getting started", leave this field blank to use the Default workspace already created during the ElectricFlow installation.

Click **OK** to save the information you entered.

After clicking **OK**, you see the Project Details page and the name of the new project adjacent to the page title.

- For our scenarios, the project name is FirstNewProject.
- Your new project name will appear on the Projects page also.



Scenario requirement: No additional requirements for this scenario. This scenario uses the *local* agent (resource) that was included on your local machine when ElectricFlow was installed.

Overview

This scenario establishes that ElectricFlow and its components were installed successfully and guides you through creating a simple procedure with a parameter that will echo Hello World. After you create this procedure, you will run the procedure and review the results.

At the end of this scenario, you will be familiar with the following ElectricFlow concepts and features:

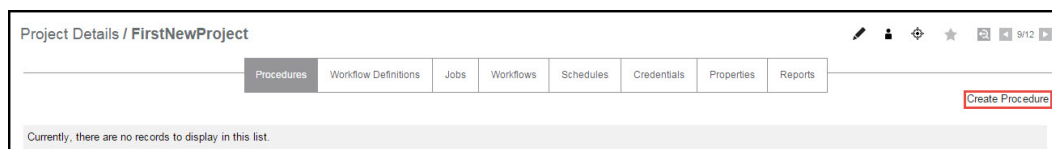
- Projects
- Procedures
- Steps
- Parameters
- Jobs
- Navigating the Job Details web page

Begin Scenario 1

Step1. Select the Projects tab

On the Projects page, you can see the default projects installed during the ElectricFlow installation and the new project you just created.

Select your new project name to go to its Project Details page, then click the **Create Procedure** link.



Step 2. Create a procedure

On the New Procedure page, you need to be concerned with the section of the page illustrated below only.

Enter the following information:

Field Name	Description
Name	For this scenario, <code>Hello World</code> is the procedure name. At a later time when you are creating your own procedures, you can choose any unique name that is most meaningful for procedures in your project.
Description	(optional) Enter a text description for your procedure if you choose to do so.
Default Resource	Use the Browse link and choose <code>local</code> . Using <code>local</code> accesses the agent installed during the ElectricFlow installation. At a later time after you have configured other resources, your Browse list could contain many choices.

Your New Procedure page will look similar to the following:

Project: FirstNewProject
New Procedure

Name:

Description:

Job Name Template:

Default Resource: [Browse](#)

Before leaving this page, notice the "breadcrumb" in the top-left corner. Most ElectricFlow web pages contain a breadcrumb for easy return to the previous page.

Click **OK** to continue.

After clicking **OK**, ElectricFlow takes you to the Procedure Details page—this is the page you use to add important information to your procedure. When your procedure contains steps, parameters, or email notifiers, and so on, those objects will be displayed here for your reference.

Step 3. Create a parameter

An ElectricFlow procedure can define one or more parameters. These parameters are similar to the parameters of a subroutine in a programming language. Parameters allow you to define a "re-usable" procedure by using symbolic variable names in place of a single, fixed value. When you run the procedure, you can type in a value to use for that particular job. You can refer to the parameter in a step for a procedure, using the standard

ElectricFlow "property reference syntax" - `$(...)`.

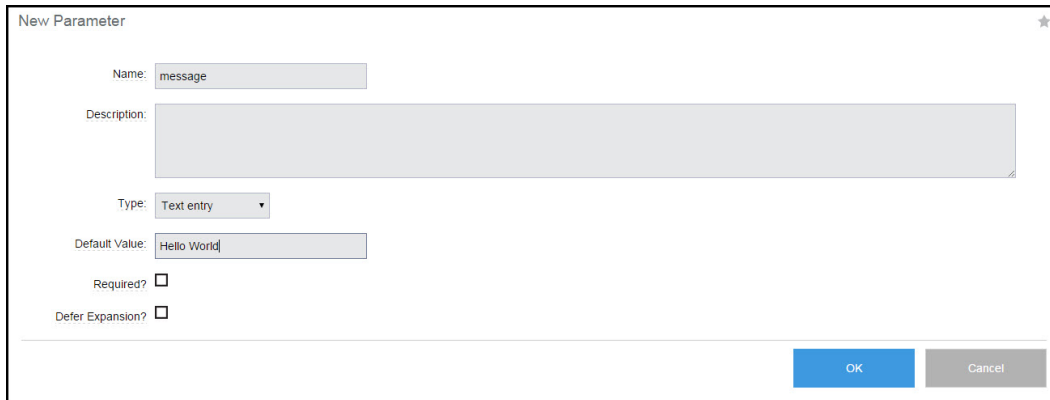
For example, see **Step 4**, specifically the text in the Command(s) text box.

On the Procedure Details page, click the **Create Parameter** link to go to the New Parameter page.

Enter the following information:

Field Name	Description
Name	Enter the parameter name <code>message</code> for this scenario.
Type	Use the default <code>Text entry</code> for the parameter type.
Default Value	Type-in <code>Hello World</code> for the default parameter value.

Your New Parameter entries will look like the following:



Click **OK** after filling in the fields and ElectricFlow returns you to the Procedure Details page.

On the Procedure Details page:

Notice the Parameters table is populated with your parameter entry.

The screenshot shows the 'Procedure Details' page for a procedure named 'Hello World'. The page has a top navigation bar with 'New Step', 'Command', 'Subprocedure', 'Plugin', and 'Custom' links. Below this, there are four main sections: 'Procedure Steps', 'Parameters', 'Email Notifiers', and 'Custom Procedure Properties'. Each section has a 'Create' button and a table of existing items. In the 'Parameters' section, a parameter named 'message' with a default value of 'Hello World' and type 'entry' is highlighted with a red box. The 'Email Notifiers' section is currently empty. The 'Custom Procedure Properties' section has one property named 'ec_customEditorData'.

Also notice the name of the procedure we created (Hello World) is shown adjacent to the page title for your reference.

Step 4. Create a step

Now we can create a step on this procedure. On the Procedure Details page, to create a New Step for our current purpose, click the **Command** link to go to the New Step page.

On the New Step page:

The following screen displays the portion of the New Step page you need for this scenario.

Notice this version of the New Step page contains a Command text box to enter the information you need. At a later time, when you choose to create a different step type, the New Step page will be populated with the appropriate fields required for that step type.

Enter the following information:

Field Name	Description
Step Name	Use <code>Print message</code> for the step name. For work outside this scenario, you can enter any unique name of your choice for the step name.
Description	(optional) Add a text description about this step if you need one.

Field Name	Description
Command(s) box	<p>Type <code>echo \${message}</code></p> <p>In this command, <code>message</code> refers to the parameter we created.</p> <p>Note: The ElectricFlow Server does not actually interpret (or even understand) the contents of this field. ElectricFlow simply expands all property references and then passes the resulting text block to the ElectricFlow Agent. The Agent copies the text block to a temporary file and then invokes the Agent's native shell command, passing the name of the temporary file as a parameter to the shell command. The overall result is the same as if you had created this block of text as a BAT file (on Windows) or a shell script (on Linux). The operating system of the Agent machine takes over from that point to actually run your commands.</p>
Resource(s)	<p>For this scenario, leave this field blank to use the default resource you set on the procedure. If multiple resources are configured, you have the option to change the resource for each step if you prefer.</p>

Your New Step page will look like the following example:

New Step

General

Name:

Description:

Command

Command(s):

echo \${message}

Resource:

Postprocessor:

Click **OK** to create the step and ElectricFlow returns you to the Procedure Details page to see its entry as displayed in the next screen example.

Procedure Details / Hello World

Procedure Steps						New Step:	Command	Subprocedure	Plugin	Custom
Step Name	Resource	Action	Parallel	Time Limit						Actions
Print message		echo \${message}								

Parameters						Create Parameter
Parameter Name	Default Value	Type	Req?	Description		Actions
message	Hello World	entry				

Step 5. Run the procedure

Notice the Run-arrow icon among the links at the top of the tables. If you click the **Run** link, the procedure will run as-is immediately.

For this scenario, hover your mouse over the adjacent down-arrow and select the **Run...** option.

Selecting this option provides the Run Procedure page where you can add or modify the parameter values as necessary.

Run Procedure / Hello World

Parameters

message:

Advanced

Priority:

Impersonation: ☒ Use pre-defined credential ☐ Use specific credential ☐ Use a specific user

The Run Procedure page has the information we want for this scenario, so click the **Run** button.

When a procedure runs, it creates a "job".

After clicking **Run**, ElectricFlow takes you to the Job Details page to see the status of the "job" and watch its running progress.

This web page supplies a wealth of other information.

Job Details / job_6_20150711160208

Completed with Success
 Start Time: 2015-07-11 16:02:08 PDT
 Elapsed Time: 00:00:00.693

Project: FirstNewProject
 Procedure: Hello World
 Launched by: admin
 Priority: normal

View: All ▾

Steps | Diagnostics | Parameters | Properties | Notifiers | Published Artifact Versions | Retrieved Artifact Versions

Expand All | Collapse All

Step Name	Log	Status	Elapsed Time	Resource	Actions
Print message		Completed with Success	00:00:00.157	local	

The Job Details page contains two sections. The section at the top of the page is the job's summary, displaying the job's status and other general information:

- The "green check" icon is displayed to show the job completed successfully.
- The "Completed with Success" section displays the result of the procedure that ran.
- The General Information section displays the project and procedure names and who launched the job.

The section below the summary area displays detailed job information, including tabs to expand information and links to access various other ElectricFlow web pages.

- Notice the "Success" entry in the Status column—this is your step result.
- In the Log column, click the icon in the Print message row to see the outcome/result of your Hello World Print message step. Your file will look similar to the following:

```
Job: job_6_20150711160208
Workspace File / Print message.db7dd766-2820-11e5-8679-0050568f747c.log
Hello World
```

- Select the Parameters tab to see the parameter you created and its value.

For complete information about the Job Details page functions and available information, click the **Help** link in the top-right corner of this ElectricFlow web page.

Scenario Extension - Adding Another Step

Each procedure you create can contain as many steps as you need. We created only one step in this scenario. This example shows how to modify a procedure by adding additional steps. You can always add more steps or edit an existing step before you run a procedure again.

To add a step to an existing procedure:

1. On the Job Details page, select your procedure name (at the top of the page) to go to the Procedure Details page.
2. On the Procedure Details page, click the **Command New Step** link again.
3. On the New Step page, enter information in the fields as follows:

- **Name**—Print Step Info
- **Command(s)**—Enter the following text:

```
print "This step was launched by ${launchedByUser} on the resource
${/myResource/resourceName}.\n";
```
- **Shell**—ec-perl
- Click **OK** to save the step and return to the Procedure Details page.

Note: You can move this new step to the top of the procedure by hovering your mouse over the icon next to the step name, then drag-and-drop the step to the top of the list. This is a useful function particularly if you find you have created steps in a different order in which you need them.

Click **Run** (from the Procedure Details page).

Finding an ElectricFlow Automation Platform Web Page

If you "lose" the Job Details page or any other ElectricFlow web page, several navigation methods are available:

- Use your browser's Back button if you were recently on a page you want to revisit.
- Use ElectricFlow "breadcrumbs." On some pages you see breadcrumbs on the left-side of the page that note your current position/level within the project.
- Use the tabs at the top of the page to drill-down to the page you need.

For example:

- **To find the Job Details page:**
 Select the Jobs tab and notice that your procedure name is now listed in the Job column, accompanied by a date, job number, and other related job information on the same row. When you run a procedure, it becomes a job. Select the job you want to see and you will go to the Job Details page for that job.
- **To find the Procedure Details page:**
 Select the Projects tab, then select your project to go to the Project Details page to see a list of procedures in that project. Click on a procedure to go to the Procedure Details page for that procedure.

Note: A "project" is a container for all related procedures you create. You can create as many projects as you need, using the **Create Projects** link on the Projects page.

As you use ElectricFlow, you will quickly learn other navigation paths for your particular interests. In addition, most ElectricFlow web pages contain a "star" icon in the upper-right corner. Click this icon to add that page to your Home page for quick "one-click" access in the future.

Summary

This simple `Hello World` procedure demonstrated how to quickly create a procedure and how any command can be wrapped and parameterized using an ElectricFlow procedure. If you have an existing script you want to automate, you can call that script within a step command block. This scenario also introduced you to running a job, entering parameters for the job, and viewing results for the job you ran.

Go to [Scenario 2](#)

Getting Started - Scenario 2 - Creating a Procedure That Uses an SCM

Scenario requirements: Before you begin this scenario, ElectricFlow needs to be able to access your Source Control Management (SCM) system.

To set up a Source Control system (SCM) to use with ElectricFlow, you need:

- a running SCM system
- an active source depot in your SCM system
- an ElectricFlow agent that can communicate with your SCM server

Note: If the machine where you installed ElectricFlow can communicate with your Source Control (SCM) server, this requirement is met. *If not*, return to the installation executable file and run the ElectricFlow Agent install program on a machine that has access to your SCM system.

To install ElectricFlow agent software on another machine, copy the installation executable file to that machine, then double-click on the file to run the installation program. Select Express Agent when the install window opens.

When installation is complete, configure this new agent as an ElectricFlow Resource. See the beginning of [Scenario 4](#) for more information.

- Configure ElectricFlow to communicate with your SCM system.
- Select the Administration tab, then the Source Control subtab.
- The Source Control Configurations page displays a table listing Source Control configurations after you create them. Click the **Create Configuration** link and the following web page is displayed.



- On the New Source Control Configuration web page, use the drop-down menu to choose your SCM type.
For our example, we chose **Perforce**.

- Depending on which SCM you choose, the page expands to display appropriate fields for entering values to configure your SCM.

See the next screen example.

Enter information in the fields as follows:

- Configuration Name - For our example, we use the name, "p4". You can choose any name for your SCM Configuration.

Note: You will need this name later, so remember the name you choose for your SCM configuration. A configuration name is important because you may want to create more than one source control configuration.

- P4PORT - For our example, we use p4:3710 to designate the P4PORT. Your specification will be different.

Important: Other fields remain blank to use the defaults, but you may need to specify some or all information in the remaining fields depending on how your internal systems and access are set up for access to your SCM system.

New Source Control Configuration

SCM Type: Required

Configuration Name: Required

Description:

Login As: User Name: Password: Retype Password:

P4PORT (host:port):

P4TICKETS:

P4CHARSET:

P4COMMANDCHARSET:

P4HOST (override):

Debug: ☒

OK Cancel

Click **OK** after filling-in the fields.

For additional help with this ElectricFlow web page, click the **Help** link in the top-right corner of the web page.

Overview

This scenario guides you through creating a common build process integrated with your SCM system and build utility. And you will learn how to navigate and modify the procedure definition using the ElectricFlow Procedure Details web page.

At the end of this scenario, you will be familiar with the following ElectricFlow concepts and features:

- Source control configuration
- Creating more complex steps
- Creating a build process
- Navigating the Procedure Details web page

Begin Scenario 2

Step 1. Select the Projects tab

On the Projects page, select the project name you created in Scenario 1 (in the first column) to go to the Project Details page, then select the **Create Procedure** link to begin.

Step 2. Create a procedure

On the New Procedure page, enter a Procedure Name - it is **required**.

You can use any unique name you choose—**do not** use the same procedure name you chose for Scenario 1. Procedure names within the same project must be unique.

- Name - For this scenario, we use `Basic Build` for the procedure name.
- Default Resource - Use `local` for the resource name.

Click **OK** to go to the Procedure Details page.



Step 3. Create a parameter

On the Procedure Details page (illustrated above), click the **Create Parameter** link to go to the New Parameter page.

- **Name** - For our example, the parameter name is `target`. You can choose any unique name of your choice for your parameter name—parameter names must be unique within a procedure.
- **Default Value** - We chose `jar` for this value.

Click **OK** to continue and to return to the Procedure Details page to see the parameter you just created.

Step 4. Create a step to “checkout” source files from your SCM system

From the same Procedure Details page, click the **Plugin** link to create a New Step.

- In the Choose Step dialog, select **Source Code Management** from the left pane.
- Next, select **ECSCM-Perforce** from the list, then select “Perforce - Checkout” from the right pane.
- Click **Close** to go to the New Step page, which will be populated with the fields you need to enter for the Perforce SCM.
 - Both the Subprocedure and the Parameters sections are populated with fields to enter information so ElectricFlow can communicate with your Perforce SCM system, using the ElectricFlow Perforce plugin.

Note: Depending on the SCM you chose to work with, including Perforce, the New Step screen you see may be different than the following example. If you need additional help configuring your SCM, see the corresponding SCM plugin on the Plugin Manager page. Each plugin has its own Help topic.

New Step

General

Name: checkout

Description:

Command

Subprocedure

Subprocedure: ECSCM-Perforce : CheckoutCode

Change

Resource:

Parameters

Configuration: default

Source Type: Client Template

Sync Type: Standard Sync

Client Template: default

Workspace Name Postfix:

Destination Directory: project

Changelist (or Label):

Generate changelog report: ☒

Last Snapshot:

Updates File:

Parallel Sync: ☐ threads=10

Forced Sync: ☐

Required

Required

Required

Required

Enter information in the fields as follows:

Field Name	Description
Name	Enter a unique name for your subprocedure step. For this scenario, the step name we supplied is <code>checkout</code> .
Description	(Optional) Enter a text description for this step.
<div><div>In the Subprocedure section:</div><div>This section displays the subprocedure name or a plugin name because the plugin is being called as subprocedure.</div><div>Note: The plugin name defined: "EC" is for Electric Cloud, "SCM" identifies the plugin as belonging to the Source Control Management category, and "Perforce" identifies the name of the source control system.</div><div>Our example illustrates the fields for the Perforce SCM plugin. Your fields will be different if you are using a different SCM. These parameter fields request some of the same values you already specified when you created the Source Control Configuration.</div></div>	
ECSCM-Perforce	Clicking this link takes you to the Project Details page for the plugin. Generally, and for this scenario, you do not need to access this page to make changes—it is for your reference only.

Field Name	Description
CheckoutCode	Clicking this link takes you to the Procedure Details page for this procedure. Generally, and for this scenario, you do not need to access this page to make changes—it is for your reference only.
Change	Clicking this link displays a dialog that allows you to change the project, plugin, or subprocedure. If you make a change in the Change Subprocedure dialog, the Subprocedure and Parameters sections on the New Step page will update automatically with corresponding fields for your new choice. Note: All SCM plugins that can checkout code contain the <code>checkoutCode</code> procedure.
Resource	The <code>local</code> resource was specified earlier. If this field remains blank, the <code>local</code> resource is used.
Note: The following two parameter fields are common among all SCM plugins:	
Configuration	The name of your Source Control Configuration.
dest	The destination where you want to put your checked-out code—it is the directory within your job workspace. Each job adds a new directory to the workspace. At this point, you have not defined multiple workspaces, so the ElectricFlow default workspace is used.

For more information about workspaces, see the "[Workspaces and Disk Space Management](#)" Help topic.

Click **OK** after entering values and to continue to the Procedure Details page, which should now look similar to the following example:

The screenshot shows the 'Procedure Details / Basic Build' interface. It features two main sections: 'Procedure Steps' and 'Parameters'.

Procedure Steps: This section has a table with columns: Step Name, Resource, Action, Parallel, Time Limit, and Actions. A new step named 'checkout' is added, with the Action 'ECSCM-Perforce CheckoutCode'. The 'checkout' text in the Step Name column is highlighted with a red box.

Parameters: This section has a table with columns: Parameter Name, Default Value, Type, Req?, Description, and Actions. A new parameter named 'target' is added, with a Default Value of 'jar' and a Type of 'entry'.

The Procedure Details page now displays the name of our new procedure, Basic Build, and its new `checkout` step using the Perforce SCM, and the `target` parameter with the `jar` value.

Step 5. Create a step to run the build

This step can be another subprocedure step, a command step, or a custom step. Any procedure can contain multiple types of steps, any of which may be either simple or complex.

Steps can contain or use:

- parameters and/or properties
- many of your existing scripts, including shell scripts

For this step example (to run the build), we will create an `Ant` step. To invoke Ant, you could write a script that calls Ant directly, or you could call the ElectricFlow Ant plugin to invoke the `runAnt` procedure. For this example, we will use the ElectricFlow Ant plugin.

- From the Procedure Details page, click the **Plugin** link to see the Choose Step dialog and select the Build category.
- Select the EC-Ant plugin to see and select the "Ant - Run Ant" step.
- When the New Step page is displayed, notice the Subprocedure section is populated with the chosen EC-Ant plugin, and the Parameters section provides the appropriate fields for this Ant step.

New Step

General

Name:

Description:

Command

Subprocedure

Subprocedure: EC-Ant : runAnt [Change](#)

Resource:

Parameters

Ant Location:

Build File:

Libraries:

Properties:

Property File:

Debug: ☐

Diagnostics: ☐

Output Level: ☒ Normal
☐ Quiet
☐ Verbose

Log File:

Target:

Additional Commands:

Postp Line:

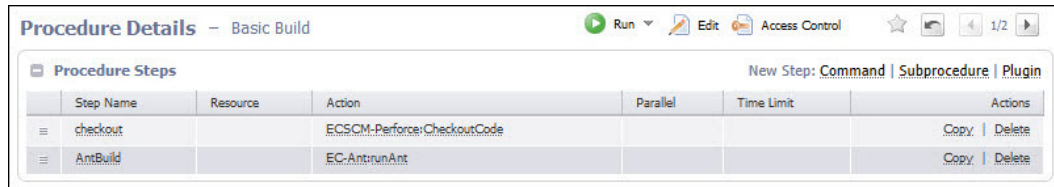
Enter information in the fields as follows:

Field Name	Description
Name	For this example, we specified <code>AntBuild</code> for the step name. You can use any unique name you choose for this step.
Subprocedure section: Note the EC-Ant plugin project name and the <code>runAnt</code> procedure name.	
Resource	Leave this field blank to continue using the local resource.
Parameters section:	
Build File	Use <code>build.xml</code> .
Target	For this example, we used <code>\${target}</code> .
Working Directory	For this example, we used <code>project</code> .

For this scenario, we leave the remaining fields blank, but you may need to enter additional information depending on your Ant invocation.

Click **OK** to continue and return to the Procedure Details page.

The following screen example illustrates the two steps we created for this scenario.



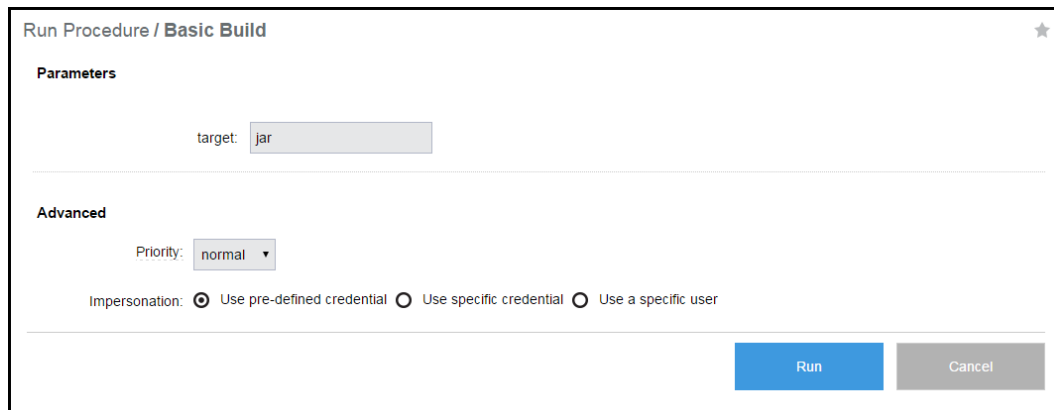
Procedure Details - Basic Build						
New Step: Command Subprocedure Plugin						
Step Name	Resource	Action	Parallel	Time Limit	Actions	
checkout		ECSCM-Perforce:CheckoutCode			Copy	Delete
AntBuild		EC-Ant:runAnt			Copy	Delete

Step 6. Running the procedure

On the Procedure Details page, hover your mouse over the small down-arrow on the right-side of the **Run** link, choosing **Run...**

- **Run Immediately** - This option runs the procedure immediately as set.
- **Run...** - This option displays the Run Procedure web page where you may alter any existing parameters for this procedure, or set an existing credential.

Note: This scenario does not introduce you to the Advanced section of the Run Procedure page. For more information on these topics, click the **Help** link in the top-right corner of the Run Procedure page.



Run Procedure / Basic Build

Parameters

target:

Advanced

Priority:

Impersonation: ☒ Use pre-defined credential ☐ Use specific credential ☐ Use a specific user

Click **Run** to run the procedure and continue to the Job Details page to see the status of your running job.

Step 7. On the Job Details page

The Job Details page now displays the status and results of running the Basic Build procedure.

Your Job Details page will look somewhat different if you supplied other values in this scenario.

Job Details / job_7_20150711170511

Completed with Success
Start Time: 2015-07-11 16:02:08 PDT
Elapsed Time: 00:00:00.693

Project: FirstNewProject
Procedure: Basic Build
Launched by: admin
Priority: normal

Steps
Diagnostics
Parameters
Properties
Notifiers
Published Artifact Versions
Retrieved Artifact Versions

View: All
Expand All | Collapse All

Step Name	Log	Status	Elapsed Time	Resource	Actions
checkout		Completed with Success	00:00:01.293		
checkoutMethod		Completed with Success	00:00:01.293		
runMethod		Completed with Success	00:00:01.397	local	
AntBuild		Completed with Success	00:00:01.482		
createCommandLine		Completed with Success	00:00:01.210	local	
runCommandLine		Completed with Success	00:00:00.060	local	

Records per page: 100
1 thru 6 of 6

More about the Job Details page

- The General Information section at the top of the table provides links back to the **Project > Procedure** that was launched to create this job.
- Click on a Step Name to go to the Job Step Details page.
- Click the Parameters tab to see parameters in this job.
- If there are errors, the Diagnostics tab has specific information.

More about the Procedure Details page

To go to the Procedure Details page at any time:

- Select the Projects tab.
- Select the Project name (that contains the procedures you want to see) to go to the Project Details page.
- Select a Procedure name to go to the Procedure Details page.

You may want to review to the Procedure Details page frequently because it displays the steps that make up your procedure, the parameters contained in the procedure, email notifiers, and so on. Full edit capabilities are available by selecting any object (step, parameter, and so on).

Also, you can run this procedure again “on demand” by clicking the **Run** link at the top of the page.

For more information about the Procedure Details page functions, click the **Help** link in the top-right corner of the Procedure Details page.

Scenario extension - Add a step to show workspace contents

If you are curious about what your current default workspace contains at this point, you can create a step to see the workspace.

- Select the Projects tab, select FirstNewProject (or your project name if you used something different), select the Basic Build procedure name.
- Click the **Command** link to create a New Step—a new *command* step.
- The New Step page is displayed with a Command text box.
 - For Name, use `seeWorkspace`.
 - In the Command(s) text box:
 - for Windows - type `dir /s`
 - for Linux - type `ls -r`
- Click **OK** to create the step and go to the Procedure Details page.
- Click **Run** to create the job and to go to the Job Details page.

When the job is completed, click the log icon in the Log column to see the contents of the current workspace for this procedure. Your Workspace File will be similar to the following example:

Job: job_7_20150711170511

Workspace File – seeWorkspace.1375.log

Volume in drive N has no label.
Volume Serial Number is EAB8-74CE

Directory of N:\job_7_20150711170511

07/11/15	09:43 AM	<DIR>	.
07/11/15	09:43 AM	<DIR>	..
07/11/15	09:43 AM	<DIR>	project
07/11/15	09:43 AM		2,770 runMethod-checkoutCode-1377.log
07/11/15	09:43 AM		0 seeWorkspace.1375.log
		2 File(s)	2,770 bytes

Directory of N:\job_7_20150711170511\project

07/11/15	09:43 AM	<DIR>	.
07/11/15	09:43 AM	<DIR>	..
07/11/15	09:43 AM		1,141 build.xml
07/11/15	09:43 AM		338 Makefile
07/11/15	09:43 AM		338 Makefile2
07/11/15	09:43 AM	<DIR>	src
		3 File(s)	1,817 bytes

Directory of N:\job_7_20150711170511\project\src

07/11/15	09:43 AM	<DIR>	.
07/11/15	09:43 AM	<DIR>	..
07/11/15	09:43 AM		144 HelloWorld.java
		1 File(s)	144 bytes

Total Files Listed:
6 File(s) 4,731 bytes
8 Dir(s) 244,679,413,760 bytes free

Summary

This scenario demonstrated how easily ElectricFlow can integrate with your SCM system and build utilities.

Go to [Scenario 3](#)

Getting Started - Scenario 3 - Notification, Scheduling, and Reporting

Scenario requirements: For this scenario, you need to set up an Email Configuration to use the email notification feature.

To set up an Email Configuration:

- Select the Administration tab, then select the Email Configurations subtab.
You will see a blank Email Configurations page. As you create one or more email configurations, they

will be listed in table format on this page.

- Click the **Add Configuration** link.

On the New Email Configuration page, enter information in the fields to define your mail system.

Field Name	Description
Name	Enter a unique name for your email configuration.
Description	(Optional) Enter a text description for your reference.
Mail Protocol	Use the drop-down arrow to make a selection.
Mail Host	Enter the name of your mail host.

Enter information in the remaining fields according to the specifications you need.

Your New Email Configuration page should look similar to the following example:

The screenshot shows a web form titled "New Email Configuration" with a star icon in the top right corner. The form contains the following fields and controls:

- Name:** A text input field containing "TestEmailConfig". To its right is the label "Required".
- Description:** A large text area with a vertical scrollbar.
- Mail Protocol:** A dropdown menu currently showing "SMTP".
- Mail Host:** A text input field containing "hostname.yourcompany.com". To its right is the label "Required".
- Mail Port:** A text input field containing "25".
- Mail From:** A text input field containing "yourgroup@yourcompany.com". To its right is the label "Required".
- Mail User:** A section containing three sub-fields:
 - User Name:** A text input field containing "admin".
 - Password:** A password input field with masked characters (dots).
 - Retype Password:** A password input field with masked characters (dots).

At the bottom right of the form are three buttons: "Test", "Save", and "Cancel".

Click **Save** after filling-in the fields or click the **Test** button first if you want to make sure your email configuration works.

Enter an email address in the pop-up box and click **Send**.

Note: For additional help with this ElectricFlow page, click the **Help** link in the top-right corner of the page.

Overview

This scenario guides you through creating an email notification, invoking your build process with a timed (standard) schedule, invoking your build process using a continuous build integration trigger, and introduces ElectricFlow reporting features.

At the end of this scenario, you will be familiar with the following ElectricFlow concepts and features:

- Creating an email configuration
- Creating an email notifier
- Creating schedules
- Creating reports

Begin Scenario 3

Step 1. Creating an email notifier

On the Procedure Details page for the Basic Build procedure, click the **Create On Completion Email Notifier** link to begin. This link takes you to the New On Completion Email Notifier page.

Procedure Details – Basic Build

Run Edit Access Control 1/2

Procedure Steps New Step: Command | Subprocedure | Plugin

Step Name	Resource	Action	Parallel	Time Limit	Actions
checkout		ECSCM-Perforce-1.1.11.40415:CheckoutCode			Copy Delete
AntBuild		EC-Ant-1.0.3.40250:runAnt			Copy Delete
seeWorkspace		dir /s			Copy Delete

Parameters Create Parameter

Parameter Name	Default Value	Type	Req?	Description	Actions
target	jar	entry			Delete

Email Notifiers Create On Start Email Notifier **Create On Completion Email Notifier**

Currently, there are no records to display in this list.

Two email notifier types for jobs or job steps:

- On Start Notifier—Sends an email when a job or job step starts.
- On Completion Notifier—Sends an email when a job or job step completes.

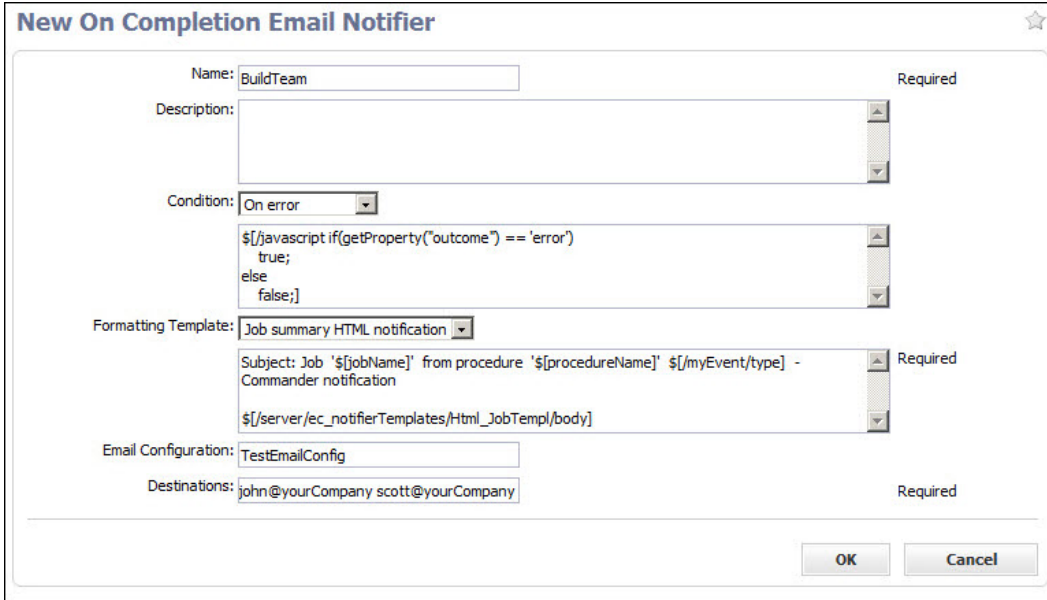
The fields for each of these notifiers are similar, but we chose the On Completion notifier because you may find you use this one more frequently.

On the New On Completion Email Notifier page:

Enter information in the fields as follows:

Field Name	Description
Name	For this example, we use <code>BuildTeam</code> . This name can be an arbitrary text string.
Description	(Optional) Enter a text description for your reference. ElectricFlow does not use or interpret this information.
Condition	Use the pull-down menu to select the type of condition you need for this email notifier. Edit the auto-supplied condition in the text box or add a completely new script for your purpose. The condition specifies whether the notifier should send a message depending on the result of a property expansion. If the result is empty, non-zero, or "true", the message is sent. If the result is "0" or "false", the message is not sent.
Formatting Template	<p>For this example, choose the "Job summary HTML" notification template, or you can use the drop-down menu to select from a list of global, ready-to-use formatting templates. Depending on the type of email notifier you are creating, the available template choices in the drop-down menu will be different.</p> <p>To customize your template, edit the auto-supplied text in the Formatting Template text box, or you can add a completely new template for your purpose. Note: Any edits made in this text box will not be saved to the global template, and the template will appear as a Custom template when you return to this notifier definition. Additionally, make sure the content is formatted correctly, i.e., no illegal characters or spacing.</p>
Email Configuration	Use <code>TestEmailConfig</code> , the email configuration we created.
Destinations	This is a space-separated list of valid email addresses, email aliases, or ElectricFlow user names, or a property reference that expands into such a list.

After filling-in the fields, your page may look similar to the following example:



New On Completion Email Notifier

Name: Required

Description:

Condition:

`$(javascript if(getProperty("outcome") == 'error')
true;
else
false;)`

Formatting Template:

Subject: Job '[jobName]' from procedure '[procedureName]' \$[/myEvent/type] - Commander notification
\$[/server/ec_notifierTemplates/Html_JobTempl/body]

Email Configuration: Required

Destinations: Required

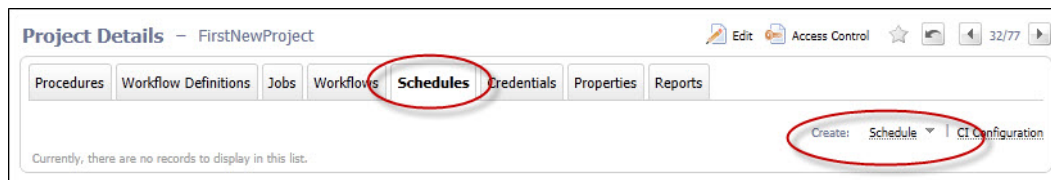
Click **OK** to create your email notifier and to return to the Procedure Details page, where you can now see the Email Notifier you created.

Note: For more information, see the [Email Notifiers](#) Help topic.

Step 2. Creating a standard schedule

To create a standard "timed" schedule:

- Select the Projects tab.
- Select your project name to go to the Project Details page.
- Select the Schedules subtab, then select the down-arrow to the right of the "Schedule" link, and choose "Standard Schedule" to go to the New Schedule page.



On the New Schedule page:

The following is an example of the New Schedule page.

- Notice the "breadcrumb" above the New Schedule page title—this entry displays the name of the project that will contain this schedule.

- After entering a name (unique within your project) for your schedule (we chose "Sentry-MainBuild"), choose the procedure your schedule will run.
 - Click the **Change** link adjacent to the Procedure field.
 - In the pop-up menu, leave the Current project selected.
 - Place your cursor in the Procedure field to see a list of procedures in your project.
 - Select the procedure you want this schedule to run. We chose "Basic Build".

Project: FirstNewProject

New Schedule

General

Name:

Sentry-MainBuild

Procedure:

FirstNewProject : Basic Build

Change

Description:

Parameters

target:

jar

Frequency

Run at 00:00.

Days:

☒ Every Day

☐ Once

☐ Days of Week

☐ Days of Month

☐ Custom

Repetition:

Run Once

at

(hh:mm)

Advanced

Enabled:

☒

Misfire Policy:

Ignore the missed triggers and wait for the next scheduled time

Time Zone:

[- Default -]

Priority:

normal

Impersonation Credential

Credential Project:

☒ Current

☐

Browse

Credential Name:

Browse

OK

Cancel

Enter information in the remaining fields as follows:

Field Name	Description
Description	(optional) A plain text or HTML description for this object. If using HTML, you must surround your text with <html> ... </html> tags. The only HTML tags allowed in the text are: <a> <div> <dl> <i> <p> <pre> <style> <table> <tc> <td> <th> <tr>

Field Name	Description
Parameters section If the procedure has parameters, enter their values in this section. The procedure cannot execute unless all required parameters are provided.	
Frequency section "Run at" is a summary section: As you make selections for the following Days and Repetition sections, a summary of what you selected appears here.	
Days	Select weekdays, month days, or create a custom frequency when you want the schedule to run. Note: The "day" applies to the start time if the time range spans two days, that is, if the time range crosses the midnight boundary.
Repetition	If Run Once - the schedule will run once per selected day, at the time you specify. If Run Every - the procedure will execute repeatedly during the specified time range. For example, if you select a time range from 11:00-14:00 and an interval of 40 minutes, the procedure will execute 5 times on each of the selected days: at 11:00 am, 11:40 am, 12:20 pm, 1:00 pm, and 1:40 pm. If Run Continuously - as soon as one job ends, the scheduler will trigger immediately to run the job again.
Advanced section	
Enabled	If this box is "checked," the schedule will run. You can disable this schedule whenever necessary.

Note: For this scenario, you do not need to enter information for any remaining fields. Later, when creating an actual timed schedule for one of your procedures, refer to the Help topic for this page for more information.

Click **OK** to save your information and create this schedule.

At any time, you can return to the Project Details page for this project, select the Schedules subtab, and then select the Schedule Name from the table to go to the Edit Schedule page to modify this schedule.

Step 2a. Creating a continuous integration build trigger

At a later time when you have procedures and steps created to do the work you need, you may prefer implementing continuous integration build configurations for your software builds. Unlike a standard schedule

that runs on days and times you specify, a continuous integration build can be triggered to run each time code is checked into your source code management (SCM) system.

Adding a project to the Continuous Integration Dashboard is quick and easy. Each project can have one or more CI configurations, depending on the number of source branches you want to monitor. After the simple configuration process, you can visually see status and other information about your continuous integration jobs as builds are triggered to run.

To find the Continuous Integration Dashboard, select the Home tab, then select the Continuous Integration subtab. The dashboard will be empty until you start adding projects. For help using the Continuous Integration Manager and Dashboard, click the **Help** link in the upper-right corner of the dashboard web page.

Step 3. Creating a report

ElectricFlow provides multiple reports and custom report capabilities to help you manage your build environment.

Summary of available report types:

- Real-time reports - filtered view of your ElectricFlow data in real-time
- Build reports - summary reports produced at the end of a build and attached to the job
- Batch reports - summaries of your build environment with trends over time, two types:
 - Default Batch reports - automatically installed during ElectricFlow installation and scheduled to run daily (Cross Project Summary, Variant Trend, Daily Summary, Resource Summary, Resource Detail)
 - Optional Batch reports - you can configure and schedule these reports to fit your requirements (Category, Procedure Usage, Count Over Time, or Multiple Series reports)
- Custom reports - your choice to create and add at any time

Note: Batch and Custom reports must be run on the ElectricFlow server agent (local agent) only. These reports use the BIRT report engine, which is installed only on the ElectricFlow server.

Defining a “saved” search

Before you create a report, you need to define a “search” to focus the report on the information you want to see.

- Select the Search tab to go to the Define Search page.

Enter information in the fields as follows:

Field Name	Description
Number of results	For this example, do not alter the default settings in these fields as you see them. These fields do not affect the search you are creating.
Results Per Page	
Object Type	For this example, choose Job for this search.

Field Name	Description
Intrinsic Filters section	Click the Add Intrinsic Filter link one or more times as needed to further define the sort criteria for your search. We selected Project Name from the drop-down menu and typed-in our project name, then selected Procedure Name and typed-in our <i>Basic Build</i> procedure name.
Custom Filters section	Not needed for this scenario.

Your Define Search page will look similar to the following example:

Define Search

General
Number of Results:
Results Per Page:
Object Type:

Intrinsic Filters

Project Name

equals

FirstNewProject

Remove

Procedure Name

equals

Basic Build

Remove

+ Add Intrinsic Filter

Custom Filters

+ Add Custom Filter

OK

Cancel

Click **OK** after filling-in the fields and to go to the Search Results page.

Job Search Results ☆

9 Results for "Project Name" equals "FirstNewProject" and "Procedure Name" equals "Basic Build" New Search Save Filter Edit Search Refresh Search

Job	Status	Priority	Procedure	Launched By	Elapsed Time	Start Time	Actions
job_1250_201104121322	✓ Success	normal	FirstNewProject:Basic Build	admin	00:00:14.124	2011-04-12 13:22:11 PDT	Delete
job_1249_201104121321	✓ Success	normal	FirstNewProject:Basic Build	admin	00:00:12.260	2011-04-12 13:21:56 PDT	Delete
job_1248_201104121321	✓ Success	normal	FirstNewProject:Basic Build	admin	00:00:12.297	2011-04-12 13:21:37 PDT	Delete
job_1247_201104121321	✓ Success	normal	FirstNewProject:Basic Build	admin	00:00:13.525	2011-04-12 13:21:17 PDT	Delete
job_1209_201104121015	✓ Success	highest	FirstNewProject:Basic Build	project: Electric Cloud	00:00:14.965	2011-04-12 10:15:19 PDT	Delete
job_1086_201104081645	✓ Success	normal	FirstNewProject:Basic Build	admin	00:00:18.129	2011-04-08 16:45:14 PDT	Delete
job_1084_201104081644	✓ Success	normal	FirstNewProject:Basic Build	admin	00:00:14.537	2011-04-08 16:44:24 PDT	Delete
job_1079_201104081623	✓ Success	normal	FirstNewProject:Basic Build	admin	00:00:14.499	2011-04-08 16:23:25 PDT	Delete
job_806_201104060942	✓ Success	normal	FirstNewProject:Basic Build	admin	00:00:12.045	2011-04-06 09:42:49 PDT	Delete

Records per page: 20 1 thru 9 of 9

Click **Save Filter** to see the Save Filter dialog box.

In this dialog box, use the drop-down arrow to select the project name for your report, then enter a name of your choice for the filter. We used `BasicBuildFilter` for the filter name to tie the filter name to the Basic Build procedure for which we are creating a filter.

Save Filter

Project Name: FirstNewProject ▼

Filter Name: BasicBuildFilter

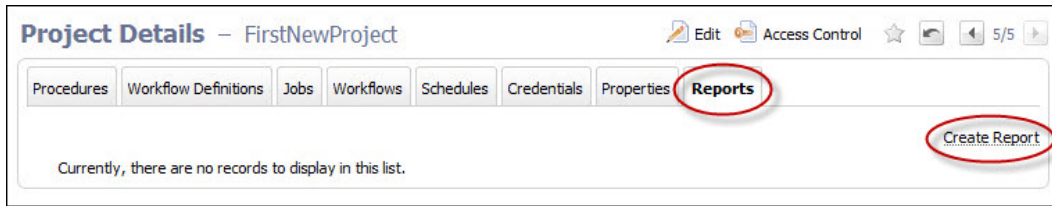
OK
Cancel

Click **OK** to return to the Job Search Results page—now we are ready to define the report we want to see.

Define the report

For this scenario, we will create an Optional Batch report. Select the Projects tab, select *your* project name to go to the Project Details page, then select the Reports subtab.

Click the **Create Report** link.



On the Optional Batch Reports page:

For this scenario, select the Multiple Series report tab.

Enter information in the fields or select the appropriate values as follows:

Field Name	Description
Report Title	This is your report title. Type over the default report name, choosing any unique name for your report. We supplied <code>Basic Build Report</code> for our report name.
Saved Filter	Project - Click your mouse inside this field to see a list of projects from which to make a selection. We select our <code>FirstNewProject</code> project name. Filter - Use the <code>BasicBuildFilter</code> we created earlier.
Time Period	Use the drop-down menu to select the time period.
Create thumbnail?	Check this box so you can view this report on your Home page. (We will create a "thumbnail" report view on your Home page at the end of this scenario.)
Object Type	Use the drop-down menu to select <code>Job</code> for this example.
Table column choices	
Chart Type	Use the drop-down menu to choose the chart type.
Function	Use the drop-down menu to choose the function you need.
Property Name	Use the default property value or delete the text and click your mouse in the blank field to see a list of possible properties. We chose "outcome" for this example.
Display Name	You can choose a different unique Display Name if you prefer to do so.
Stacked	Select this check box to see your report results "stacked" versus overlaid.
The "X" icon	Click this icon to delete any row you no longer need.

Field Name	Description
Add Series button	Select this button if you would like to create additional table entries for additional report information.
Chart Options	Use the down-arrow to adjust the Time Grouping and see the defaults for the X and Y axis.

Your Multiple Series report page will be similar to the following example:

- Run Report button - (information only for this scenario) this button takes you to the Job Details page to run the report immediately—one time only.
- Click the **Create Schedule** button - this button displays a box with a default schedule name you can change.

- Enter a name for the schedule—again, we tied the schedule name to the procedure name for which we want the report.
- Click **OK** to go to the Project Details page.

Project Details – FirstNewProject Edit Access Control ☆

Procedures Workflow Definitions Jobs Workflows **Schedules** Credentials Properties Reports

Create: [Schedule](#) | [CI Configuration](#)

Schedule	Enabled	Procedure	Priority	Description	When to run	Time Zone	Actions
Basic Build Report Schedule	<input checked="" type="checkbox"/>	EC-Reports:RunReport_MultiSeries	normal		Run at 02:00.	America/Los_Angeles	Run Copy Delete
Sentry-MainBuild	<input type="checkbox"/>	Basic Build	highest		Run at 00:00.	America/Los_Angeles	Run Copy Delete

Now you can see the report you just created listed in the Schedules table.

Click the **Run** link in the Actions column to run this report and then go to the Job Details page.


On the Job Details page , you will see your report running.

When the report is generated, you will see a Links section in the summary section at the top of the Job Details page.

This section contains a link to the report you generated.

Click this link to see your report.

Job Details – EC-Reports-1.1.3.36678-RunReport_MultiSeries-1281 Run Again Delete ☆ >>



Completed with Success
 Start Time: 2011-04-12 15:46:08 PDT
 Elapsed Time: 00:00:48.979

General Information



Project: [FirstNewProject](#)
 Procedure: [EC-Reports-1.1.3.36678:RunReport_MultiSeries](#)
 Schedule: [Basic Build Report Schedule](#)
 Launched by: [admin](#)
 Priority: normal

Links

[Basic Build Report](#)

Steps Diagnostics Parameters Properties Notifiers

View: All Expand All Collapse All

Step Name	Log	Status	Elapsed Time	Resource	Actions
RunCustomReport		 Success	00:00:47.387	local	Edit

For more information about the Job Details page, click [here](#).

For more information about ElectricFlow reports, see the [Reports](#), [Creating Custom Reports](#), and other report Help topics.

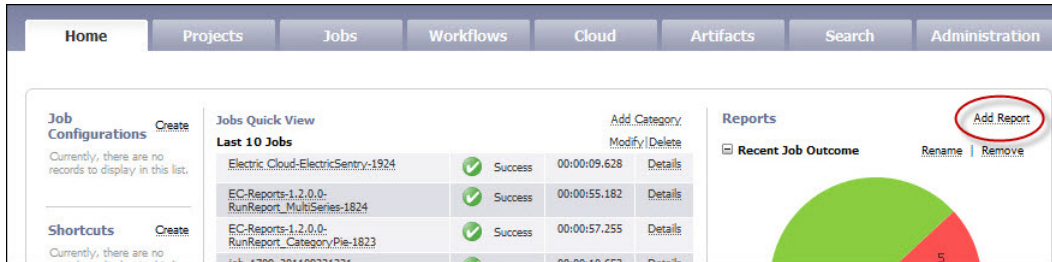
Scenario extension - Add a thumbnail report to your Home page

When we created a report, we "checked" the Create thumbnail? checkbox, so we are ready to activate the thumbnail report view on the Home page.

Note: Viewing an interesting report assumes there is ample data to report. While you can see a report after running one job, it is not very interesting unless you can see trends or comparisons after running numerous jobs.

To see a thumbnail report on the Home page:

- Select the Home tab.
- Click the **Add Report** link and a New Report page is displayed.



On the New Reports page:

New Report

Project:

FirstNewProject

Required

Report:

Basic Build Report

Required

Title:

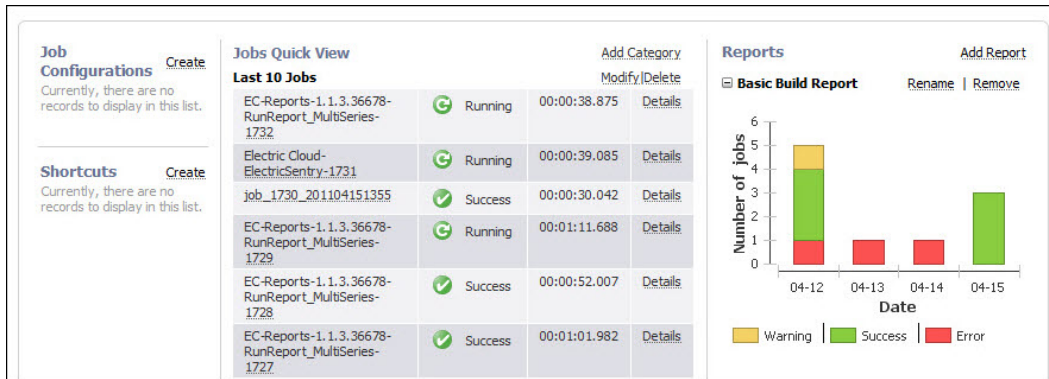
Basic Build Report

Required

- Project - Use the down-arrow to select the project containing the report you want to view—FirstNewProject.
- Report - This is the report name you entered in the Report Title field when you configured this report—Basic Build Report.
- Title - This is the report title you want to see on your Home page to identify this report. We used the same Basic Build Report name.

Click **OK** to generate the thumbnail report view and to go to the Home page.

The following screen example illustrates how your Home page may look with the thumbnail report.



For more information about how to use the Home page, see [The Home Page](#) Help topic.

Summary

This scenario demonstrated how to create an Email Configuration so you could use an email notifier, how to create a continuous integration schedule, and how to set up a basic report with a "thumbnail" report view on your Home page.

Go to [Scenario 4](#)

Getting Started - Scenario 4 - Multi-agent Build and Test

Scenario requirements: For this scenario, you can use the configurations you created for Scenario 2 and 3. However, before you begin this scenario, you need to configure an additional agent machine. If you have a firewall running on the ElectricFlow server machine, disable it now or allow access to ports 8000, 8443, 61613, 80, and 443.

Installing ElectricFlow Agent software:

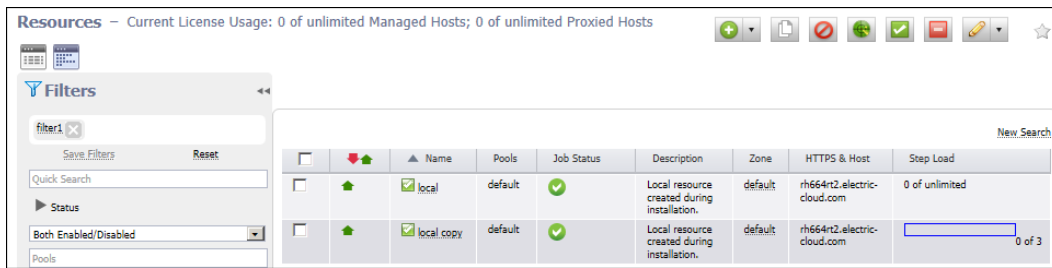
Install ElectricFlow Agent software on the machine you intend to use as an agent to run builds. For the simple purpose of getting started, install this agent on a different machine with the same operating system as the ElectricFlow server.

- Copy the ElectricFlow installation executable file to the machine you intend to use as an ElectricFlow agent.
- Double-click the executable file to begin the installation program.
- When the installation program opens, select the Express Agent installation.

Note: During the agent installation, you will have the opportunity to create a resource and provide a resource name. Enter any name you choose for your new resource. For this scenario, we will name our new resource, `Resource2`.

- After agent installation, verify that your new resource information is included in the ElectricFlow resource table.

- Select the Cloud tab to go to the Resources page.



Note: At a later time when you are more familiar with ElectricFlow, you can use the Resources page to install additional agents. See the Resources page Help topic, then specifically review the "Install or Upgrade Remotes Agents" section.

To configure a shared workspace:

For Linux

If your agents are installed on Linux, change the default workspace to point to a network location accessible to both machines:

- Select the Cloud tab, then select the Workspaces subtab, to see a list of workspaces.
- Select the workspace named "default" to edit the default workspace.
- Change the value of the UNIX Path to a network location accessible to the ElectricFlow server and agent machines.
- Make sure this location is readable and writable by the users chosen during the server and agent installations.

After fulfilling these scenario prerequisites, you are ready to proceed.

For Windows

If using a Windows environment, the shared workspace is created for you already.

Overview

This scenario guides you through creating a multi-agent build and test process integrated with your SCM system, build utility, and unit/system tests. You can invoke this build using a schedule, Continuous Integration (CI), or "on demand". At the end of this scenario you will be familiar with the following ElectricFlow concepts and features:

- Agent-only installation
- Copying a procedure to create a new procedure
- Running steps in parallel
- Pools

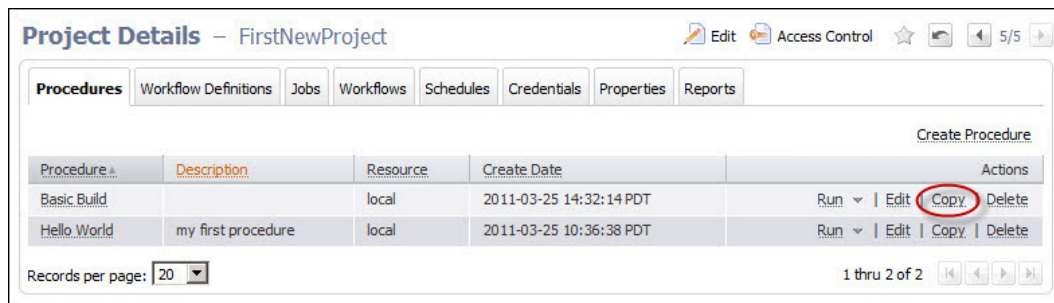
- Post processor
- Search

Begin Scenario 4

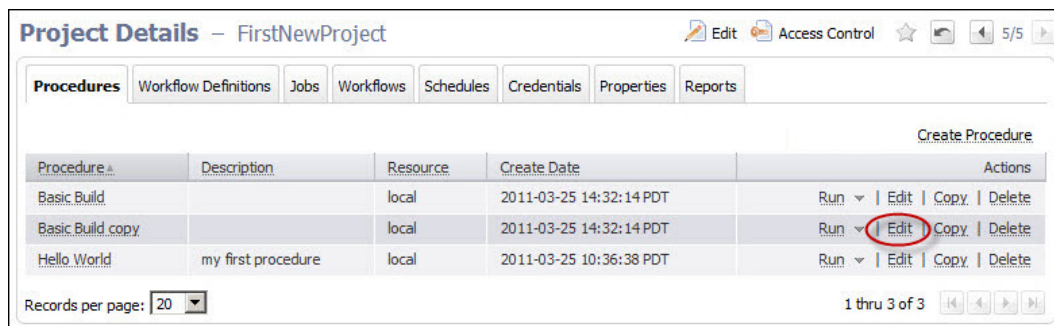
Step 1. Create a new procedure

This time, we are going to *copy* an existing procedure, rename it, and add another step.

- Select the Projects tab, then select the FirstNewProject project name.
- On the Project Details page, click the **Copy** link to copy the Basic Build procedure.



- When the Basic Build copy procedure is displayed, you can click its name to go to the Procedure Details page to verify it contains the same contents of the Basic Build procedure.
- Return to the Project Details page.



- Select the **Edit** link for the Basic Build copy procedure.

On the Edit Procedure page:

- Name—Enter a new procedure name, `MultiAgentBuild`.
- Description—Add a description noting this procedure was copied from the Basic Build procedure.
- Default Resource—Select your new resource, `Resource2`, for this procedure.

Click **OK** after entering your information and ElectricFlow will take you back to the Project Details page.

Your Project Details page should now look similar to the following example:

Project Details – FirstNewProject [Edit](#) [Access Control](#) [Star](#) [Refresh](#) [5/5](#)

Procedures Workflow Definitions Jobs Workflows Schedules Credentials Properties Reports

Create Procedure

Procedure	Description	Resource	Create Date	Actions
Basic Build		local	2011-03-25 14:32:14 PDT	Run Edit Copy Delete
Hello World	my first procedure	local	2011-03-25 10:36:38 PDT	Run Edit Copy Delete
MultiAgentBuild	copied from the Basic Build procedure	Resource2	2011-03-25 14:32:14 PDT	Run Edit Copy Delete

Records per page: 20 1 thru 3 of 3

This page now displays the new MultiAgentBuild procedure with a description, and a new resource is designated for this procedure to use.

Step 2. Create two new Ant steps

- Beginning on the Project Details page, click the MultiAgentBuild procedure name to go to the Procedure Details page.
- Click the AntBuild step **Copy** link, twice.

Procedure Details – MultiAgentBuild [Run](#) [Edit](#) [Access Control](#) [Star](#) [Refresh](#) [3/3](#)

Procedure Steps [New Step: Command](#) | [Subprocedure](#) | [Plugin](#)

Step Name	Resource	Action	Parallel	Time Limit	Actions
checkout		ECSCM-Perforce-1.1.11.40415:CheckoutCode			Copy Delete
AntBuild		EC-Ant-1.0.3.40250:runAnt			Copy Delete
AntBuild copy 2		EC-Ant-1.0.3.40250:runAnt			Copy Delete
AntBuild copy		EC-Ant-1.0.3.40250:runAnt			Copy Delete
seeWorkspace		dir /s			Copy Delete

- Now you see two additional steps names: AntBuild copy and AntBuild copy 2.
- Click the AntBuild copy step name to go to the Edit Step page.
- Notice the AntBuild copy step name is adjacent to the page title because this is the step we are going to edit.

Modifications for creating the new Ant steps:

Because this is an "Edit" page, some of the fields already contain the information we originally supplied to create the first Ant step. The following list contains the fields we need to edit or add new information:

- Name**—For this example, we chose `unittest` for the step name. Type over the existing text to edit the text.
- Resource**—Enter the name, `TestPool`, because we will be creating a resource pool for testing. At this point, you need to enter a pool name only (no spaces in the name).
- Target**—Edit this field to: `unittest`

In the Advanced section:

- Run in Parallel—Click the check box to select this option. You are designating that you want this step to run in parallel with one or more other steps.

Click **OK** to create the first of two new Ant steps and ElectricFlow returns you to the Procedure Details page.

Edit Step — AntBuild copy
Access Control

General

Name:
Description:

Subprocedure

Subprocedure: EC-Ant : runAnt [Change](#)
Resource:

Parameters

Ant Location:

Build File:

Libraries:

Properties:

Property File:

Debug: ☐

Diagnostics: ☐

Output Level:
☐ Normal
☐ Quiet
☐ Verbose
☒ normal

Log File:

Target:

Additional Commands:

Postp Line:

Working Directory:

Environment Variables:

Advanced

Precondition:

Run Condition:

Error Handling: Procedure continues, but overall status will be error

Time Limit: minutes

Run in Parallel: ☒

From the **Procedure Details** page, click the AntBuild copy two step name to go to the Edit Step page again.

To create the next Ant step, make the same edits (above) as you did to create the `unittest` step, except:

- Name—Change the step name to `systemtest`.
- target—Change this field to: `systemtest`

Click **OK** to create the second of two new Ant steps and to return to the Procedure Details page. You now have three Ant steps, including two new Ant steps, `unittest` and `systemtest`, with `TestPool` as the designated resource.

Step Name	Resource	Action	Parallel	Time Limit	Actions
checkout		ECSCM-Perforce-1.1.11.40415:CheckoutCode			Copy Delete
AntBuild		EC-Ant-1.0.3.40250:runAnt			Copy Delete
unittest	TestPool	EC-Ant-1.0.3.40250:runAnt	✓		Copy Delete
systemtest	TestPool	EC-Ant-1.0.3.40250:runAnt	✓		Copy Delete
seeWorkspace		dir /s			Copy Delete

Also notice, the Parallel column now contains "checkmarks" for the 2 steps we want to run in parallel.

Step 3. Create a resource pool

Select the Cloud tab to see the following Resources page with a table listing the currently available resources.

Name	Pools	Job Status	Description	Zone	HTTPS & Host	Step Load
local	default	✓	Local resource created during installation.	default	rh664n2.electric-cloud.com	0 of unlimited
Resource2	default	✓	Local resource created during installation.	default	rh664n2.electric-cloud.com	0 of 3

To put these two resources in a resource pool that we previously "named" when we created the new Ant steps, we need to edit each resource.

Click the `local` resource name to go to the Resource Details panel, then click the **Edit** link at the top of the panel.

On the Edit Resource panel:

The only change we need to make is to add the `TestPool` pool name to the Pool(s) field.

Edit Resource

Name:

Description:

Agent Host Name:

Agent Port Number:

Default Workspace:

Pool(s):

Default Shell:

Step Limit:

Artifact Cache Directory:

Zone:

Repository Names:

Connection Type:

Enabled: ☒

OK **Cancel**

Click **OK** to save your change and return to the Resources page.

- Notice the Pool(s) column now has TestPool specified as a resource pool where the `local` resource is a member.
- Repeat this process to put `Resource2` in the TestPool resource pool.

A resource can belong to one or more pools. If any resource belongs to multiple pools, all pools would be listed in this column. For more information about the Resources page, click [here](#) or click the **Help** link in the top-right corner of the web page.

The following screen example should be similar to your Resources page.

Resources – Current License Usage: 0 of unlimited Managed Hosts; 0 of unlimited Proxied Hosts

Filters

filter1 X Save Filters Reset

Quick Search

Status

Both Enabled/Disabled

Pools

		Name	Pools	Job Status	Description	Zone	HTTPS & Host	Step Load
<input type="checkbox"/>	+	local	TestPool	✓	Local resource created during installation.	default	rh664rt2.electric-cloud.com	0 of unlimited
<input type="checkbox"/>	+	Resource2	TestPool	✓	Local resource created during installation.	default	rh664rt2.electric-cloud.com	0 of

Step 4. Run the procedure

To run the new MultiAgentBuild procedure, select the Projects tab > FirstNewProject project name > MultiAgentBuild procedure.

On the Procedure Details page:

- Click the **Run** link at the top of the page.
- Click **Run** on the Run Procedure page.

Your Job Details page should be similar to the following example.

Job Details – job_352_201106151426

Abort Run Again Delete Save Configuration Access Control

Running with Success

Start Time: 2011-06-15 14:26:56 PDT
Elapsed Time: 00:00:48.956

General Information

Project: FirstNewProject
Procedure: MultiAgentBuild
Launched by: admin
Priority: normal

Steps Diagnostics Parameters Properties Notifiers

View: All Expand All Collapse All

Step Name	Log	Status	Elapsed Time	Resource	Actions
checkout		Running	00:00:47.869		Abort Edit
checkoutMethod		Running	00:00:47.875		Abort Edit
runMethod		Running	00:00:47.882	Resource2	Abort Edit
AntBuild			00:00:00.000	Resource2	Abort Edit
unittest			00:00:00.000	TestPool	Abort Edit
systemtest			00:00:00.000	TestPool	Abort Edit
seeWorkspace			00:00:00.000	Resource2	Abort Edit

Records per page: 100 1 thru 7 of 7

Note: The screen example above illustrates a job that is still running—notice the green circular-arrow icon at the top of the page. This icon changes to a "checkmark" if this job completes successfully.

Scenario extension - Using postp

ElectricFlow implements data collection with a postprocessor. The postprocessor is a command associated with a particular procedure step. If the postprocessor is specified for a step, it executes concurrently with the main step command. The postprocessor runs on the same machine as the main command and in the same working directory, and it retrieves the log file from the step as its standard input.

The standard ElectricFlow postprocessor is called **postp**. **postp** scans the step's log file looking for interesting output such as error messages and then sets properties on the job step to describe what it found. For example, postp might create a property named "errors" whose value is the number of error messages in the log file, or a property named "tests" that counts the number of tests the step executes. Also, postp can extract portions of the step log that contain useful diagnostic information and save this information for reporting.

When you create a step, you can specify `postp`. If you already have a step and want to add postp reporting, you just have to edit the step to include postp.

- From the previous Job Details page screen example, click the **Edit** link for the AntBuild step.
- On the Edit Step - AntBuild page, in the Parameters section, add `postp` to Postp Line field.

Edit Step - AntBuild Access Control ☆

General

Name:

Description:

Subprocedure

Subprocedure: [Change](#)

Resource: [Browse](#)

Parameters

Ant Location:

Build File:

Libraries:

Properties:

Property File:

Debug: ☐

Diagnostics: ☐

Output Level: ☒ Normal
☐ Quiet
☐ Verbose

Log File:

Target:

Additional Commands:

Postp Line:

Working Directory:

Click **OK** to save your edit and return to the Job Details page.

Run this job again to see the postp result—click the **Run/Run Again** link.

On the next screen example, notice the AntBuild step now shows "3 compiles" in the Status column. Using postp supplies additional information.

Job Details – job_1934_201104181440
 Run Again Delete Save Configuration >>

Completed with Success
 Start Time: 2011-04-18 14:40:08 PDT
 Elapsed Time: 00:00:40.900

General Information
 Project: FirstNewProject
 Procedure: MultiAgentBuild
 Launched by: admin
 Priority: normal

Steps | Diagnostics | Parameters | Properties | Notifiers

View: All Expand All | Collapse All

Step Name	Log	Status	Elapsed Time	Resource	Actions
[-] checkout		✓ Success	00:00:13.069		Edit
[-] checkoutMethod		✓ Success	00:00:13.066		Edit
runMethod		✓ Success	00:00:13.158	local	Edit
[-] AntBuild		✓ Success	00:00:14.186		Edit
createCommandLine		✓ Success	00:00:05.284	local	Edit
runCommandLine		✓ 3 compiles	00:00:08.237	local	Edit
[-] unittest		✓ Success	00:00:06.999		Edit
createCommandLine		✓ Success	00:00:06.100	local	Edit
runCommandLine		✓ Success	00:00:00.447	local	Edit
[-] systemtest		✓ Success	00:00:05.311		Edit
createCommandLine		✓ Success	00:00:04.319	local	Edit
runCommandLine		✓ Success	00:00:00.419	local	Edit
seeWorkspace		✓ Success	00:00:00.356	local	Edit

Records per page: 100
 1 thru 13 of 13

For more information about using postp, see the ["Postprocessors"](#) Help topic.

Summary

This scenario demonstrated how ElectricFlow can drive a complete build and test process—including building steps to run in parallel and creating a resource pool. This scenario also provided a brief look at how you can use ElectricFlow to execute rapid root cause analysis by reviewing postp diagnostics.

Automation Platform Setup

The topics in this section describe how to set up the automation platform UI.

After completing the tasks, you can create, configure and manage the objects needed to automate your build-test processes, application deployment, and pipelines.

Automation Platform Home Page

[Job Configurations](#)

[Shortcuts](#)

[Jobs Quick View](#)

[Reports](#)

Overview

This page provides a convenient console for running jobs and viewing results.

- This web page is your ElectricFlow Dashboard for tracking project health.
- You can customize the page to display only the jobs that interest you and to provide one-click access to the jobs you run frequently. Each of the Home page sections can be customized.
- The Reports section allows you to see thumbnail report images that are updated each time the report is generated.
- This page provides shortcuts for quick access to pages you use most frequently in the automation platform or anywhere on the web.

IMPORTANT: If you are using or plan to use the EC-Homepage plugin to share your Home page sections, you will not see some links normally available to you. For example, if a particular Jobs Quick View category is shared, you **cannot** Delete or Modify that category. You can perform these functions only if the category belongs to you alone.

For information on sharing your Home page configuration, see the "[Home page configurations](#)" section in the "[Customizing the ElectricFlow Platform UI on page 583](#)" Help topic.

Job Configurations

Procedures in ElectricFlow can contain complex sets of parameters— making it difficult to remember the correct parameters for a particular situation and tedious to re-enter those parameters every time the procedure is invoked. Job Configurations provide a one-click solution to this problem. When you create a job configuration, you enter all the information needed to run a procedure, including parameters and/or a credential. All job configurations are displayed here on the Home page. Invoke a particular configuration by clicking its name in the Job Configurations section.

Create Job Configurations three ways

- On the Home page, create a job configuration from "scratch" by clicking the **Create** link in the Job Configurations section.
 - In the Create Configuration pop-up menu, select the project and procedure you want to use for creating this configuration.
- From the Job Details page for a previously invoked job, click the **Save Configuration** link at the top of the page.

Your saved job configuration is displayed on your Home page.
- From the Edit Schedule page, click the **Save Configuration** link at the top of the page.

Your saved configuration is displayed on your Home page.

Shortcuts

Use shortcuts to save frequently visited ElectricFlow web pages, so those pages are immediately accessible. You can create a shortcut to any page on the web also.

Create Shortcuts two ways

- Mouse-over the "star" icon at the top of any automation platform page and click "Add current page" to add that page to the shortcut list. Mouse-over the star icon again and click "Remove current page" to remove the shortcut for that page. The star icon is yellow for pages saved as a shortcut and gray for those pages not saved as a shortcut.
- Click the **Create** link in the Shortcut section and provide a name and URL to create a shortcut.

To modify or update a shortcut, click the **Edit** link adjacent to the shortcut you want to change.

Shortcuts can be accessed conveniently from any automation platform web page. Mousing-over the star icon displays a list of shortcuts saved by the current user. Click on a shortcut name to view the page.

Note: The Shortcut section may contain static entries that cannot be deleted. And if you "share" your Home page, the Edit link will not be available for shared items.

Jobs Quick View

Perhaps only a few jobs on this server are of interest to you. For example, you may care about jobs you launch manually and official builds for the products you work on, but you may not care about production builds for other products or personal jobs for other users.

The Jobs Quick View allows you to define job categories that are interesting to you.

The Home page displays the most recent jobs in each category, and you can easily click-through to get more details about any of those jobs. For example, clicking on a job name takes you to that job's Job Details page.

Create a job category

- Click the **Add Category** link in the Jobs Quick View section.
After creating a category, results are displayed on the Home page. Clicking the **Details** link displays a summary to the right of the category. In addition to job status and other diagnostic information, the summary displays running steps and failed steps (containing errors and warnings).
- Click the **Modify** link to edit the Jobs Quick View category or the **Delete** link to remove that job from the Jobs Quick View category.
- Click on the **Details** link to see job summary information—the summary remains visible, regardless of mouse location, until you click somewhere else on the page.

Note: If you "share" your Home page, the Modify and Delete links will **not** be available for shared items.

Reports

You can configure reports you would like to see on a regular basis and display a "thumbnail" report graphic in this section.

- Click the **Add Report** link to go to the New Reports page.
After filling-in the information on the New Reports page, you will see a thumbnail view of your report

(after it runs) on your Home page.

Note: If the drop-down menu on this page is empty, click the **Help** link on the New Report page for more information.

- Click the Collapse/Expand (+/-) box to see the full thumbnail report or just the report title.
- Click the **Rename** link if you need to rename your report.
- Click the **Remove** link if you need to remove this report from your Home page.

Note: If you "share" your Home page, the Rename and Remove links will not be available for shared items.

Customizing the ElectricFlow Platform UI

You can customize the ElectricFlow platform UI to get a more intuitive, task-specific user interface.

- [Customizing parameters](#)
- [Customizing tab layouts](#)
- [Home page configurations](#)

Customizing parameters

Two types of objects contain formal parameters – procedures and state definitions. For these objects, you can customize the way their formal parameters are presented. You can reorder the parameters and set the labels, form element types, default values, tooltip Help text, and whether or not the parameter is required. When defined, this customization shows up whenever the parameters are displayed—running a procedure, creating a step to call a procedure, creating a transition definition to a specific target state, and so on.

How do you customize parameters?

Create a property named `ec_parameterForm` on the object containing formal parameters (the procedure or state definition). The value of this property is an XML-style specification of form elements that map to the parameters. This property must be in sync with the formal parameters, which means all formal parameters must have a corresponding XML element, with no "extra" XML elements.

Custom parameter form contents

The value of the property in XML: Under a top-level element called `<editor>`, you need an element called `<formElement>` for each parameter. Under a `<formElement>`, you can specify the following tags:

- `property*` - the name of the formal parameter
- `type*` - `entry|textarea|select|radio|checkbox|project|savedSearch|credential`
- `label*` - displayed next to the element
- `value` - the initial value of the element (does not apply to credentials, which must be specified at run time)
- `required` - if true, the user must enter a value for the element
- `documentation` - text displayed in the tooltip when the form element is "moused" over

- numRows - valid for `textarea` elements; represents the height of the element
- option - a single option for `select|radio` elements; at least one is required
 - name* - the text displayed in the option
 - value - the value of the option
- checkedValue - valid for `checkbox` elements; the value of the element when it is checked
- uncheckedValue - valid for `checkbox` elements; the value of the element when it is unchecked
- initiallyChecked - valid for `checkbox` elements; whether or not the element is "checked" by default

Note: An asterik (*) in the list above indicates a *required* tag.

Example

The following example is a parameter form that rearranges parameters and sets the labels, descriptions, default values, and form elements for each parameter.

```
<editor>
  <formElement>
    <label>One:</label>
    <property>one</property>
    <documentation>The first parameter.</documentation>
    <type>entry</type>
    <value>Test value</value>
  </formElement>
  <formElement>
    <label>Two:</label>
    <property>two</property>
    <documentation>The second parameter.</documentation>
    <type>textarea</type>
    <required>1</required>
  </formElement>
  <formElement>
    <label>Three:</label>
    <property>three</property>
    <documentation>The third parameter.</documentation>
    <type>select</type>
    <option>
      <name>ABC</name>
      <value>abc</value>
    </option>
    <option>
      <name>XYZ</name>
      <value>xyz</value>
    </option>
    <value>xyz</value>
  </formElement>
  <formElement>
    <label>Four:</label>
    <property>four</property>
    <documentation>The fourth parameter.</documentation>
    <type>radio</type>
    <option>
      <name>First Option</name>
```

```

        <value>123</value>
    </option>
    <option>
        <name>Second Option</name>
        <value>456</value>
    </option>
    <value>123</value>
</formElement>
<formElement>
    <label>Five:</label>
    <property>five</property>
    <documentation>The fifth parameter.</documentation>
    <type>checkbox</type>
    <checkedValue>true</checkedValue>
    <uncheckedValue>false</uncheckedValue>
    <initiallyChecked>1</initiallyChecked>
    <value>true</value>
</formElement>
<formElement>
    <label>Six:</label>
    <property>six</property>
    <documentation>The sixth parameter.</documentation>
    <type>project</type>
    <value>myProject</value>
</formElement>
<formElement>
    <label>Seven:</label>
    <property>seven</property>
    <documentation>The seventh parameter.</documentation>
    <type>savedSearch</type>
    <value>/projects/myProject/ec_savedSearches/mySavedSearch</value>
</formElement>
<formElement>
    <label>Eight:</label>
    <property>eight</property>
    <documentation>The eighth parameter.</documentation>
    <type>credential</type>
</formElement>
</editor>

```

Customizing the tab layout

Overview

A "view" defines the layout of tabs in the ElectricFlow web UI. One or more tab views may be defined at the server, group, and user level. The default view the user sees when they first log in can be set at the server, group, and user level. The default defined on the user takes precedence over its groups and the groups take precedence over the server. Electric Cloud provides a system default view that will always show up in the user's list. Views can inherit from each other and add/modify/remove/reposition tabs and subtabs as needed.

If you define a custom view, you have to manually add tabs such as Continuous Integration so that they will appear in your custom view.

View definition syntax

A view is an XML property. The following is a list of elements that can be defined in a view:

- **tab** - a top level tab
 - **label** - the text to display for the tab
 - **url** - the target URL of the tab
 - **accesskey** - the keyboard shortcut to access this tab (a single letter)
 - **position** - the position of this tab relative to the other tabs (first tab is 1, second tab is 2, and so on)
 - **show** - when inheriting from another base view, if 0, the tab is not visible, if 1 it is visible (default 1)
 - **tab** - a subtab displayed within this tab
 - **label** - the text to display for the subtab
 - **url** - the target URL of the subtab (relative to ElectricFlow base)
 - **position** - the position of this subtab relative to the other subtabs (integer greater than 1)
 - **show** - when inheriting from another base view, if "0", the tab is not visible, if "1" it is visible (default 1)
- **base** - the name of a view from which to inherit tab and subtab definitions

The following is a basic definition of a two-tab view where one of the tabs has three subtabs.

```
<?xml version="1.0" encoding="utf-8"?>

<view>
  <tab>
    <label>Home</label>
    <url>home.php</url>
    <accesskey>h</accesskey>
  </tab>
  <tab>
    <label>Administration</label>
    <url>workspaces.php</url>
    <accesskey>a</accesskey>
    <tab>
      <label>Workspaces</label>
      <url>workspaces.php</url>
    </tab>
    <tab>
      <label>Directory Providers</label>
      <url>directoryProviders.php</url>
    </tab>
  </tab>
</view>
```



```

        <tab>
            <label>Licenses</label>
            <url>licenses.php</url>
        </tab>
    </tab>
</view>

```

The following is a view that inherits from the above view and:

1. Changes the URL of the Home tab
2. Adds a new tab called Projects and places it at the beginning of the list
3. Changes the accesskey of the Administration tab
4. Moves the Workspaces subtab to the end of the list
5. Hides the Licenses subtab

```

<?xml version="1.0" encoding="utf-8"?>
<view>
    <base>firstView</base>
    <tab>
        <label>Home</label>
        <url>customizedHome.php</url>
    </tab>
    <tab>
        <label>Projects</label>
        <url>projects.php</url>
        <accesskey>p</accesskey>
        <position>1</position>
    </tab>
    <tab>
        <label>Administration</label>
        <accesskey>z</accesskey>
        <tab>
            <label>Workspaces</label>
            <position>3</position>
        </tab>
        <tab>
            <label>Licenses</label>
            <show>0</show>

```

```
</tab>

</tab>

</view>
```

Storing views

Tab views are stored in the server, group, and user property sheets. The view definitions are stored in a property sheet called `ec_ui/availableViews`. The name of the view is set to the property's description if defined, otherwise it is set to the property's name. The value of the view property is the XML definition document described above.

Default views

The property `ec_ui/defaultView`, if set, determines the default view for users inheriting from that object.

- If this property is set on the server, it is the default for all users.
- If this property is set on a group, all users belonging to that group will see that view, overriding the server's default if it is set also.
- Finally, a user can explicitly set their view by defining the property on their own property sheet. If the user belongs to multiple groups that define defaults, then ElectricFlow chooses the first group alphabetically.

The user can set their view by clicking on their name in the navigation bar, then clicking on the **Edit Settings** link. The list of views shown here comprise all views defined in the `ec_ui/availableViews` property sheets for the server, groups to which the user belongs, and the user itself.

If views have the same name at different levels, then the user overrides the group which overrides the server. If a user selects a view and chooses to Save, then the `ec_ui/defaultView` property on their sheet is set accordingly.

Two special values are always available in this list: "EC Default" and "Inherit". The former is the default tab layout defined in the ElectricFlow web UI. The latter means to use the first view in the list. By default, all users inherit their tab view.

For more information on user settings, see the [Edit User Settings](#) Help topic.

Developing and troubleshooting

If you create an invalid view definition on a group or server property sheet, you will break the UI for all users who inherit that view. The best practice for developing tab views is to use a "test user". Store the view definition in the `ec_ui/availableViews` property sheet for that user, and set the `ec_ui/defaultView` for that user to the name of the new view. When the view is working properly and ready for other users, you can migrate the new view to a group or to the server.

If you continue to have difficulty and tab definitions are not working correctly and can no longer get to the user settings page, revert to the ElectricFlow default by running the following command:

```
ectool setProperty /users/$USERNAME/ec_ui/defaultView ec_default
```

When you log out and then log back in, you will have the default view again.

Note that changes to views are not visible until the next time a user logs in.

Home page configuration

The automation platform Home page is your configurable Dashboard. This page allows you to manage shared configurations and you can customize this page for your work preferences also. Refer to [The Home page](#) Help topic for details on creating Job Configurations, Shortcuts, Jobs Quick View, and Reports.

Shared Home page configurations can be set globally or for specific groups of users only. The general model for creating shared configurations is:

1. Set up your personal configuration the way you want it to appear to others.
To get to the configuration page, select the Administration tab > Plugins > and click the **Configure** link for EC-Homepage.
2. Click **Save** to make the configuration available to a specific group or globally to everyone on the server.

If you published the configuration to a specific group, no further action is needed. If you published the configuration globally, continue to the next step.

3. Copy and paste the Tab XML string into a specific user's or group's view.

This adds the public configuration to that user's or group's own home page and no further action is required for the user or group. If this step is not completed, additional action is required as described in [Installing the Home page](#) on page 589.

You can alter an existing shared configuration by loading from a previously saved location, make changes, then save the changes back to the original or an alternate location.

User settings

Use the Backup Settings action to save and restore your personal Home page configuration.

Select **Create** to back up and temporarily set aside your personal settings while you create or update shared settings.

Select **Restore** to retrieve your personal settings after updating shared settings.

Location

Select Global to share the new Home page with everyone on the server, or select Group to share with specific users only.

Installing the Home page

If a globally available Tab XML string (from the Configure EC-Homepage page) has **not** been pasted to the user's or group's view, it must be added to an existing view definition to replace the standard Home page with the one provided by this plugin. To do this, click **Load**. This replaces the current home page with the one that's publicly available.

Reconfigure Contact Support link

In the event you want to redirect the Electric Cloud Support URL (available from the Help link on each ElectricFlow web page) to your own support center, you can create a file to add to the ElectricFlow installed directory at `<ElectricFlow Install Dir>/apache/htdocs/commander`. Create a file named `config_user.php` with the following contents:

```
<?php
$config["contactSupportUrl"] = "your URL";
?>
```

Notes:

1. You must enter the value (URL) for your customer support site, replacing "your URL" in the example above. If no value is supplied, a blank window is displayed.
2. You must restart the Web Server for your change to take effect.

Configuring ElectricFlow

Use this page for quick access to configuring resources, workspaces, email configurations, your source control system, and defect tracking to communicate with ElectricFlow.

Note: If you are viewing this Help topic from the "Configuring ElectricFlow" web page, use these instructions:

Click the **Add** link for any object you want to configure and that objects configuration page is displayed.

- Each item you configure will be listed in this table for your easy reference.
- Configure objects listed in this table are linked to their respective **"Edit..."** page in the event you need to modify any values previously supplied. Select a configured object to edit its values.

The table below provides a brief description of ElectricFlow configuration tasks

Click on one of the **"Setting Up..."** links to see more information about that topic.

Setting Up Resources	Before resources can be set up, you need to know which agent machines are available to allocate to jobs. The ElectricFlow Resource web page displays all resources currently available to the ElectricFlow server.
Setting Up Workspaces	<p>ElectricFlow provides each job step with an area on the disk it can use for "working files" and results. This disk area is called a job <i>workspace</i>.</p> <p>A job step can create whatever files it needs within its workspace and ElectricFlow automatically places files, such as step logs, in the workspace. The location of the job step workspace is displayed on the Job Details web page.</p>
Setting Up Email Configurations	<p>If you do not create an Email Configuration, you will be unable to send email notifications to individuals or groups.</p> <p>If you have multiple users or groups in remote locations who use a different mail server, you will want to create additional email configurations to accommodate those locations if they need to receive notifications.</p>

Setting Up a Source Control Configuration	ElectricFlow needs to communicate with your Source Control system (SCM) if you intend to use the CI Dashboard for continuous integration schedules to start a new build based on new/modified source control checkins, or if you plan to configure Preflight builds (to build and test code changes before those changes are "committed").
Setting Up Directory Providers	ElectricFlow uses account information from multiple sources. In most cases, the primary account information source is an external LDAP or Active Directory repository: both user and group information is retrieved from the repository. Local users and groups can be defined within ElectricFlow.
Setting Up Defect Tracking	ElectricFlow uses plugins to integrate with numerous defect tracking systems. If the plugin you need for your defect tracking system is not automatically installed with ElectricFlow, see the Plugin Catalog to find the integration you need.

Web Interface Online Help System

Use the automation platform online Help system for more information. Click the **Help** link in the top-right corner of any product web page to see a specific Help topic for that page.

When the Help system opens, Electric Cloud recommends reviewing the Help table of contents. All Help folders above the Web Interface Help folder are user-guide style Help topics that provide more detailed information on each of their topics. If you generally prefer to use a command line rather than the ElectricFlow web interface, you will find complete ectool (the ElectricFlow command-line tool) commands and arguments here too.

Setting Up Resources

Before resources can be configured, you need to know which agent machines are available for allocation to jobs. The Resources page displays all resources currently available to the ElectricFlow server. This page is your Resources management center—you to create, modify, or delete resources, see resource status, pools or zones where resources reside, and much more.

Selecting the Cloud tab displays the Resources page.

What is a resource?

Note: When the ElectricFlow server and agents are installed on different hosts, make sure that the configuration for each agent specifies the Domain Name System (DNS) server.

A *resource* specifies an agent machine where steps can execute and...

- each resource has a logical name
- each resource refers to an agent machine by its host name

- several resources can correspond to the same physical host
- each resource can be assigned to one or more pools, or one zone

A pool is a group of interchangeable resources. For example, you might have a pool of Windows servers.

Naming a pool in a procedure step allows ElectricFlow to assign that step to any resource in the pool, which means ElectricFlow can choose a lightly-loaded resource and a pool makes it easy for you to change your server configuration.

- you can specify a resource or a pool name in a procedure step to execute that step on that resource/pool
- you can define custom properties for your resources or resource pools

For example, if configuration information varies from resource to resource, you can use custom properties to hold this information and then reference it from procedure steps where it is needed. Suppose you have a pool of test machines where the test hardware is in a different location on each machine. You could define a property named "testLocation" on each resource, then pass this property to procedure steps using a reference such as `$/myResource/testLocation`. This approach allows you to define procedure steps to run on any resource and automatically handle configuration differences.

ElectricFlow supports two resource types

- Standard
- Proxy

Standard Resources

This resource type specifies a machine running an installed ElectricFlow agent on one of the supported agent platforms, as specified in the *ElectricFlow Installation Guide*, chapter 2, "System Requirements and Supported Platforms."

Proxy Resources

This resource type requires SSH keys for authentication. You can create proxy resources (agents and targets) for ElectricFlow to use on numerous other remote platforms/hosts that exist in your environment. A proxy agent is an ElectricFlow agent, channeling to a proxy target.

- **Proxy agent** - This is an agent on a supported Linux or Windows platform, used to proxy commands to an otherwise unsupported platform. A proxy agent is an ElectricFlow agent, channeling to a proxy target. Proxy agents have limitations, such as the inability to work with plugins or communicate with ectool commands.
- **Proxy target** - This is a machine on an unsupported platform that can run commands via an SSH server.

When a step is running on a proxy resource, the proxy agent performs the following tasks:

- Uploads the `commandFile` to the Working Directory on the proxy target via SSH
 - Working Directory - the step runs in this directory on the proxy target, which defaults to the UNIX path to the workspace
- Creates a wrapper `sh` shell script to:
 - CD to the Working Directory

- set `COMMANDER_` environment variables that exist in the proxy agent's environment
- run the `commandFile` that was uploaded earlier
- Uploads the wrapper script to the Working Directory on the proxy target
- Runs the wrapper script on the proxy target
- Cleans up script files on the agent and proxy target

Setting up SSH keys

If you are not familiar with setting up SSH keys, go to the knowledge-based article called "Setting up SSH Keys in preparation to use a Proxy Agent" in the Electric Cloud Support Web Site:

1. Go to <https://electriccloud.zendesk.com/hc/en-us/categories/200176553-ElectricCommander>.
2. In the ElectricFlow KB section, enter **KBEC-00049** in the Search field.
3. Click **Search**.
4. Click the link to the knowledge-based article called "KBEC-00049 - Setting up SSH Keys in preparation to use a Proxy Agent" to open it.

On UNIX, the ElectricFlow agent looks in the `~/ .ssh` directory for the private and public key files to set up the SSH connection to the proxy target. Therefore, Electric Cloud recommends generating your key files in that directory.

On Windows, the ElectricFlow agent does not presume a default location. Therefore, in the proxy resource definition you must specify the key file location. Enter the following line in the resource "Proxy Customizations" field:

```
setSSHKeyFiles('c:\ foo\priv.key');
```

To create a new resource

Click the Cloud tab in the ElectricFlow web interface to go to the Resources page.

At the top right-side of the table, click the down arrow next to the plus sign and select **Create Resource** or **Create Proxy Resource** link to see the New Resource or New Proxy Resource panel.

From either panel, New Resource or New Proxy Resource, click the **Help** link in the upper-right corner of the page to access the [Resources](#) Help topic to see descriptions and other information to assist with filling-in the fields.

Gateways and zones

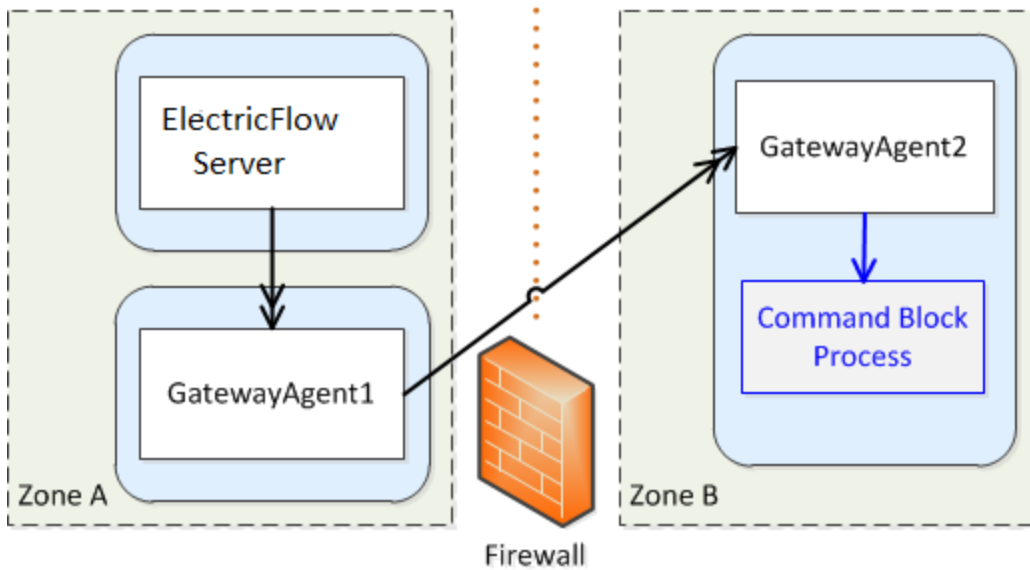
ElectricFlow provides the capability of establishing security zones for agents and repository servers.

What is a zone?

A zone consists of a collection of resources that can directly communicate with each other. Within a large corporation, a zone may encapsulate a physical site, and communication between the two sites occurs through firewalls.

Cross-zone communication

The following figure shows an example gateway and zone configuration with two zones and three machines.



Cross-zone communication is managed by proxying requests through specially marked resources called “gateway resources”. In the previous example, if the ElectricFlow server wants to run a command block process step on an agent in Zone B, it issues its request through GatewayAgent1. The request includes extra metadata that instructs GatewayAgent1 to forward the request to the target agent. The firewall is configured to allow connections from GatewayAgent1 to GatewayAgent2. It is also configured to allow connections from GatewayAgent2 to GatewayAgent1. This is necessary because API communication (CLI commands and the step completion message) is sent from GatewayAgent2 on its own outbound connection to the server (through GatewayAgent1).

Because the firewall separates two private networks, there's no guarantee that IP address ranges in one network do not overlap with those of the other. When either gateway agent wants to connect to the other, the gateway agent uses an IP address to the firewall that is valid for its side of the firewall.

What is a gateway?

The details of this connection information are recorded in a gateway object in ElectricFlow. Using the previous example as a reference, a gateway object consists of the following information:

- Name of a gateway resource in one zone (GatewayAgent1)
- Name of a gateway resource in another zone (GatewayAgent2)
- Host/port GatewayAgent2 uses to communicate with GatewayAgent1 (e.g. the firewall IP address in ZoneB)
- Host/port GatewayAgent1 uses to communicate with GatewayAgent2 (e.g. the firewall IP address in ZoneA)

Support for gateways

The previous example shows two zones for simplicity, but ElectricFlow supports an unlimited number of zones, including chained zones. For example, if a third zone called Zone C is only accessible from Zone B via GatewayAgent3 (in Zone B) and GatewayAgent4 (in Zone C), the server could issue a request to GatewayAgent1, which forwards the request to GatewayAgent2, which forwards it to GatewayAgent3, which forwards it to GatewayAgent4.

Also, for resiliency, ElectricFlow supports having multiple gateway agents between two zones. If one gateway agent goes down, the system will detect the failure and route all requests through the other gateway agent.

Setting Up Workspaces

ElectricFlow provides each job step with an area on the disk it can use for "working files" and results. This disk area is called a job workspace.

A job step can create whatever files it needs within its workspace, and ElectricFlow automatically places files in the workspace, such as step logs. Normally, a single workspace is shared by all steps in a job, but it is possible for different steps within a job to use different workspaces. The location of the job step workspace is displayed on the Job Details page for the job under "Details" for the step.

To create a new workspace

ElectricFlow can handle any number of workspaces.

- To define a workspace from the web interface, click the Cloud > Workspaces tabs.
- On the Workspaces page, click the **Create Workspace** link at the top right-side of the table and the New Workspace web page is displayed.

Click the **Help** link in the top-right corner of the web page for assistance with filling in the fields to create a workspace.

To see the Workspaces Help topic now, click [Workspace - create new or edit existing workspace](#).

After you create a workspace specification, you will see it listed on the Workspaces page.

Setting Up Email Configurations

If you do not create an Email Configuration, you cannot send Email Notifications to individuals or groups.

If you have multiple users or groups in remote locations who use a different mail server, you will want to create additional Email Configurations to accommodate those locations if they need to receive ElectricFlow notifications.

To create an email configuration

To create an Email Configuration, click on the Administration > Email Configurations tabs, then click the **Add Configuration** link to see the New Email Configuration page. Click the **Help** link in the upper-right corner of the page to see descriptions and other information to assist with filling-in the fields.

To see the Email Configurations Help topic now, click [Email Configuration - create new or edit existing email configuration](#).

After creating an Email Configuration, you will see it listed in the table on the Email Configurations web page.

Email Notifications in ElectricFlow

ElectricFlow uses email notifications at the application, application process, and process step levels (application and component). The user settings are defined and managed in the automation platform.

In ElectricFlow, email notifications work as follows:

- The notifications are triggered based on how the job finishes (the onCompletion event) and on the success or failure of the job.
- You set email notifications at the application, application process, and process step levels.
- You specify users, groups, or email addresses as recipients of the notifications.
- You can also target notifications at specific environments.
- You can enable or disable notifications at the application, application process, or process step level.

ElectricFlow provides two default templates for success and failure. You can also create new email templates to meet your needs. The templates have a name, subject, body, and type content that are stored as properties in ElectricFlow.

Setting Up a Source Control Configuration

You must configure your source control system to communicate with the ElectricFlow server if you plan to take advantage of the ElectricFlow Continuous Integration Dashboard and associated functionality and Preflight functionality—both of these features are designed to work with a number of source control (SCM) systems.

- Continuous Integration Dashboard - use this feature to create continuous integration schedules for your build environment. The Continuous Integration Manager is the front-end user interface for the ElectricSentry Continuous Integration engine.
- Preflight - use this feature to build and test developer code before it is *committed* to the code base for your product.

ElectricFlow installs (bundles) and supports numerous source control types. After creating a source control configuration, your entry will appear in the table on the Source Control Configurations web page—to see this web page, select the Administration > Source Control tabs.

Note: If the SCM you prefer is not listed here, see the Plugin Manager page (Administration > Plugins) for a list of all currently available SCM plugin integrations available for ElectricFlow. To configure a different SCM, see the Help topic associated with that plugin.

Select one of the following links to go to the source control configuration page to configure your SCM system:

[AccuRev](#)

[ClearCase](#)

[File](#)

[Git](#)[Perforce](#)[Property](#)[Subversion](#)

The Continuous Integration Dashboard has its own Help topic. For more information on ElectricSentry or Preflight builds, see their respective Help topics: [ElectricSentry](#) , [Preflight Builds](#).

Setting Up Directory Providers

ElectricFlow uses account information from multiple sources. In most cases, the primary account information source is an external LDAP or Active Directory repository: both user and group information is retrieved from the repository. Local users and groups can be defined within ElectricFlow.

To specify a directory provider

ElectricFlow includes a web page to facilitate adding your existing LDAP or Active Directory users and groups to ElectricFlow.

- To access the Directory Providers web page, select the Administration > Directory Providers tabs.
- Click the **Add Active Directory Provider** or the **Add LDAP Provider** link to add a new provider.

Depending on which **Add... Provider** link you choose, you will see either the New Active Directory Provider or the New LDAP Provider web page. From either page, click the **Help** link in the upper-right corner of the page. The Help topic describes each field on either web page so you can easily enter information in the forms to have ElectricFlow use your existing user or group account information.

To see the Directory Provider Help topic now, click [Directory Providers - create new or edit existing directory providers](#).

After adding a provider, you will see it listed in the table on the Directory Providers web page.

Enable/Disable Local ElectricFlow Users

Two server settings properties are supplied for local ElectricFlow users (both are "on" by default):

- `enableAdminUser` - if set to '0', attempts to log in as the 'admin' user are denied as if the user did not exist
- `enableLocalUsers` - if set to '0', attempts by a non-admin local user to log in are denied as if the user did not exist

These settings are stored in properties in the `/server/settings` property sheet and can be set from ectool by calling:

```
ectool setProperty /server/settings/enableAdminUser 0
```

Or, these settings can be set from the automation platform web interface:

- Select the Administration > Server tabs.
- At the top of the page, select the **Settings** link.

Note: Disabling these properties takes effect only for new sessions. Existing sessions continue to function "as-is" until the user logs out and attempts to log in again.

wrapper.conf Properties

If you need to edit the wrapper.conf file, see <http://wrapper.tanukisoftware.com/doc/english/properties.html> for complete information about property formats.

Using ElectricFlow in Your Environment

This topic addresses some common issues you may encounter as you start using ElectricFlow and offers some tips about managing your ElectricFlow projects.

[What's in a step?](#)

[Where's the script for the step?](#)

[Environment variables](#)

[When do you use subprocedures?](#)

[How do you evolve procedures?](#)

[Porting existing scripts](#)

[ElectricFlow project version control](#)

What's in a step?

One of the first questions you may ask when you start to implement your build and test processes on ElectricFlow is how to divide processes into individual steps: "Should I use only a few steps, each of which does a lot, or a large number of fine-grain steps?" Here are some factors to help you decide how many steps to use:

- **Reporting** - If you would like a separate report of success/failure for two activities, put those activities in separate steps. If you are happy to have a single report for both activities, a single step may make sense. For example, compilation and test phases should probably be in different steps because errors in the two phases will probably be handled differently. Unit tests for product components managed by different groups may make sense in separate steps; each group can watch for errors in its step.
- **Parallelism** - If you want to use ElectricFlow to run two activities in parallel, put them in separate steps. If these activities are in the same step, ElectricFlow cannot run them in parallel.

ElectricFlow parallelism works best at a coarse grain, such as running different sets of unit tests in parallel or compiling for different platforms. Electric Cloud does not recommend trying to do fine-grain parallelism with ElectricFlow, such as compiling every individual source file in a separate step—this is likely to be complicated and brittle, resulting in an enormous number of job steps, which will make it difficult to view results. If you would like to use fine-grain parallelism for compilation, Electric Cloud recommends using our ElectricAccelerator product. In this case, you would make your compilation steps

large, with as much work in each makefile as possible because ElectricAccelerator automatically subdivides the work and runs as many sub-steps as possible in parallel. The more work in a makefile, the more efficient the process becomes.

- **Resources** - If two different activities need to run on different resources, they need to be in different steps. A single step runs entirely on a single resource.
- **Conditional steps** - If you want to skip portions of your process during some jobs, but execute them during others, it probably makes sense to put those activities in separate steps and use the ElectricFlow "Run condition" mechanism to decide whether they run during each particular job. However, if you are invoking programs like *make* that already allow you to choose which actions to perform, it may make more sense to have a single large step and handle conditional behavior with those programs, rather than with ElectricFlow. For example, when running *make*, you can specify the targets to be built.
- **Setup** - You might want to have a single step at the beginning of a procedure that processes the procedure's parameters and sets up the environment for the rest of the procedure. The setup step will create a snapshot of your source code or set up a ClearCase view also. After setup, the remaining steps should be easier to write because they just use information created during the setup step.

Where's the script for a step?

Typically, each step executes a script of some type, which is processed by a command language such as *cmd* on Windows, a *shell* on UNIX, or Perl.

Two ways to handle script execution

- The first approach - Enter the script directly into the command field for the step. You can specify the language interpreter as the shell for the step.
- The second approach - Store the script in a file that is part of the source code for your project, then specify a simple command for the step that invokes the command language to process the script file.

Electric Cloud recommends using the second approach in most cases. The main advantage of this approach is that it makes your ElectricFlow procedures more robust. For example, suppose the script needs to change as your product evolves. If you have kept the script outside ElectricFlow with the source code for the product, each product version can have its own copy of the script. When you extract the source code for the product at the beginning of a job, you also extract the scripts for its steps. You can change a step's script without worrying about its impact on other versions of the product.

However, if the script for a step is stored in the step, it is more difficult to evolve the script with new product versions. You probably still need to build older product versions, so you will worry whether the new script for the step will work with older product versions. If the script does not work, perhaps you can modify the step's script to test the version being built and take different actions for each version—this process becomes increasingly more complex as the number of versions increases. Or, you can make a separate copy of the build procedure for each product version (more on this below), but this process also gets complicated as you acquire more and more procedure versions. We find it is easier to store scripts for steps with the product code, so script changes are handled in the same way as changes to the product.

Two cases where it makes sense to store the script for a step in the ElectricFlow step

- The first case is for the first step of a job.
At this point you have not extracted the source code for the product, so you do not yet have access to

any scripts stored with the source code.

- The second case is for steps with one or two commands only, or steps already specified primarily by information stored with your source code.

For example, if a step is running a *make* or *ant* command, step behavior is already specified almost entirely by a makefile or a `build.xml` file for Ant. In this case, the step's script invokes a single, relatively simple command; no need to store this in a file. If you subsequently find that the script for a step is changing frequently, consider moving it out of ElectricFlow and into a file stored with your source code.

The object is to make your ElectricFlow procedures as reusable as possible, so no change is needed with every small change to your product. Even better, organize your ElectricFlow procedures so a single procedure can be used for multiple products. To do this, store product-or version-specific information with the product, not with ElectricFlow.

Environment variables

Your build and test scripts probably depend on certain environment variables having certain values. In your existing system, you probably set those values at the beginning of your script. However, in ElectricFlow you need to set those values in each step that depends on them. The easiest way to set values is to create a short command file during the first step of the job and save it in the top-level directory of the job workspace. The command file should contain a sequence of commands to set all environment variables required by the job. Then, in each subsequent step, invoke that command file at the beginning of the step to set environment variables for that step. This approach simplifies management of your environment variables: if you need to change a variable, you change only the setup step at the beginning of the job, and the value is reflected in all of the following steps automatically.

When do you use subprocedures?

In ElectricFlow, the action for a step can invoke another procedure, passing parameters. This is a powerful tool for structuring your processes. Subprocedures tend to be used in two ways: for *encapsulating reusable processes* and for *managing concurrency*.

- *encapsulating reusable processes*, is the preferred use. If you need to perform certain activities repeatedly in different places, you can use a subprocedure for them. For example, when we build and test the ElectricFlow product, we do it on multiple platforms, but the mechanism is virtually the same on all platforms. We implement the basic build and test mechanism for one platform in a subprocedure named `BuildAndTest`. In our main procedure for production builds, we invoke `BuildAndTest` multiple times, once for each of the platforms. If the mechanism changes, we only need to change it once in `BuildAndTest` to fix all six platforms.
- *managing concurrency* - suppose you have three steps A1, A2, and A3, that must run sequentially, and two other steps B1 and B2 that must also run sequentially, but the two groups can run in parallel. The way to implement this is to put A1, A2, and A3 in one subprocedure named "A", and B1 and B2 in another subprocedure named "B". Now you can create a top-level procedure with two steps: one invoking A and the other invoking B, and mark those steps for parallel execution. As a result, A and B will run in parallel but the steps inside each subprocedure will run serially. If you placed all five steps inside the top-level procedure, there would be no way to achieve this effect.

How do you evolve procedures?

After you start using ElectricFlow for managing your build and test processes, it will not be long before you encounter the following situation:

You have a set of ElectricFlow procedures that work fine on all existing versions of the product, but you are about to change the product in a way that affects ElectricFlow procedures. For example, you might be adding a new component, or perhaps you are going to change the way the product is installed for testing, or perhaps you have restructured the product to allow more steps to run concurrently. As a result, you need to change ElectricFlow procedures for the current version. At the same time, you need older product versions to continue building as well.

In most cases, the easiest way to handle these situations is to leave the current ElectricFlow procedures alone and keep using them for your older product version. This ensures you will not "break" older versions. Make copies of the procedures that need to change, then modify the copies and use them for your new software version.

You will have one version of procedures for each significant version of the product. For example, at Electric Cloud, we incorporate the version number into the procedure names: "Master-1.0" is the version of the Master procedure used with the 1.0 release, "Master-1.1" is used for 1.1, and "Master" with no version number is used for our current development.

Another approach would be to make a copy of the entire project for each release. This method may be easier if other information in the project, such as property values, needs to evolve also.

The "copying" approach gets more and more complicated as you add more and more versions of the procedure. To minimize this complexity, try to structure your procedures so they do not have to change frequently. Some techniques you can use:

- Use parameters for things that do change frequently, such as the branch of software to build.
- Where possible, store scripts and other information [used by the procedure] as part of the source code for the project as described above, so this information is versioned automatically, along with your source code.

Porting existing scripts

You probably already have scripts you have been using to build and test software, before switching to ElectricFlow.

Over time, you will probably want to do quite a bit of restructuring to take full advantage of ElectricFlow. However, you probably do not need to do major restructuring to get started with ElectricFlow. Here are some steps you might take to "get up and running" with ElectricFlow as quickly as possible and gradually convert to make the best use of the system.

To begin, see if you can take your existing script and run it monolithically as a single step in a procedure under ElectricFlow. This will get you running and start providing some ElectricFlow benefits for resource management, error analysis, and reporting.

Next, start dividing your previous script into separate steps for ElectricFlow. Begin with the steps easiest to separate from the rest of the script, such as the commands to compile your software. One of the challenges

you face is how to pass data from one step to another. In a monolithic script, you can keep the data in variables that persist throughout the job. As you divide the script into steps, you need to figure out which variables are used only in a single step and which variables pass from step to step. For variables that pass between steps, use ElectricFlow properties to store their values. In many cases you can compute shared data values in a single step at the beginning of the job, store their values in properties on the job, then access those values read-only in later job steps.

After dividing the steps, you can do finer-grain reporting. Start using the *postp* postprocessor to scan your log files and generate statistics and diagnostics. Over time you will probably discover you have a few error and warning messages peculiar to your site, and are not captured by *postp*'s patterns. Learn how to extend *postp* with additional patterns to capture all the information that matters to you.

Now that reporting statistics are being recorded, you may wish to develop your own reports to summarize those statistics. You can easily use *ectool* to read statistics from the properties where they are stored.

Something else you can do after steps are divided: start tuning the performance of your procedures. For example, you can start marking steps to run in parallel, and you can choose resources on a step-by-step basis to finish your jobs faster and share resources more effectively.

ElectricFlow project version control

Electric Cloud recommends exporting your ElectricFlow project data at regular intervals and saving it in the same configuration management system you use for your source code (use the "export" *ectool* command to generate an XML file representing the contents of a project or procedure). This process allows you to track project changes and also allows you to "look back" at older versions if that should become desirable. Also, you should snapshot a version of ElectricFlow project data at the time of each major software release, so you can revert to the exact ElectricFlow configuration used to generate that release if necessary.

ElectricFlow Installed Tools

This reference topic can acquaint you with some tools you may not know about or provide quick access to information for tools you use already. Select from the links in the table below to go directly to information for that tool.

Tool Name	Description
eccert	A command-line tool used to manage the ElectricFlow Certificate Authority (CA) and the certificates configured in ElectricFlow Server and ElectricFlow Agent installations.
ecconfigure	A command-line tool that can change configuration values for any locally installed ElectricFlow server, web, agent, or repository service. <i>ecconfigure</i> is a more user-friendly mechanism for configuring aspects of ElectricFlow that would otherwise require manual configuration file updates. <i>ecconfigure</i> actually manipulates relevant service configuration files on your behalf.

Tool Name	Description
<code>ecdaemon</code>	A "wrapper" program that can be used to start another program from an ElectricFlow job step—the "started" program will run as a daemon process. The ElectricFlow agent uses the facilities of the underlying operating system to make sure the process runs in a separate process group on a UNIX-based system, or outside of the normal "Windows Job" grouping in a Windows system. In either case, the ElectricFlow agent does not treat the process as one it should wait for or one it should try to "kill" if ElectricFlow needs to abort the step.
<code>ecproxy</code>	A driver script with built-in support for SSH. Every major operation can be overridden by defining a Perl function in the Proxy Customization field on the New Proxy Resource panel, available from the Resources page,
<code>ecremotefilecopy</code>	Remote file copy: When ElectricFlow agents (on platforms other than Linux or Windows) run steps that create log files in a workspace the ElectricFlow web server cannot access (through Linux or Windows agents), use <i>ecremotefilecopy</i> to recreate job logs so they are visible on those ElectricFlow agents, which then enables the web server to retrieve and render those log files.
<code>ZKConfigTool</code>	A command-line tool that imports your ElectricFlow database configuration information into your ZooKeeper server.
<code>ClusterInfoTool</code>	A command-line tool that displays information on the running ElectricFlow server cluster from ZooKeeper.

eccert

A command-line tool used to manage the ElectricFlow Certificate Authority (CA) and the certificates configured in ElectricFlow Server and ElectricFlow Agent installations.

Usage

```
eccert [ options ] command [ arg ... ]
```

Commands

<code>addTrustedServer crt</code>	Add a server CA certificate to the agent's keystore.
<code>getCRL</code>	Retrieve the contents of the current certificate revocation list.

```
initAgent [ --local | --remote ] [ options ]
```

Initialize the agent keystore with a new public/private key pair.
Generates the agent certificate signing request.
If run on the server host, the certificate will automatically be signed by the server CA, and the CA certificate and the signed agent certificate are installed in the agent's keystore.
If run on a non-server host, the signing request is left in the agent directory. If CA Cert is provided, the CA certificate is installed in the agent's keystore.

`--local`
Use the local server CA to sign the agent certificate.

`--remote`
Connect to a remote ElectricFlow server to sign the agent certificate.

`--force`
Replace any existing keystore.

`--cname name`
Use the specified name as the common name (CN) in the agent certificate subject. This is normally the fully qualified domain name used by clients to connect to the agent.

`--altNames entries`
Use the specified list of entries (comma or space separated) as the `subjectAlternateNames` list in the agent certificate. Simple names are interpreted as `dns` entries. Entries may begin with `"dns:"` or `"ip:"` to indicate the type (for example, `"ip:192.168.0.1"` or `"dns:myHost"`). If no entries are specified, then reverse DNS is used to look up the registered names of the host's IP addresses.

`initCA`
Initialize the server CA. Creates a new CA key and certificate.

<code>initServer [options]</code>	<p>Initialize the server keystore. Creates and signs the server certificate. Installs the CA certificate and the signed server certificate into the server's keystore.</p> <p><code>--force</code></p> <p>Replace any existing keystore.</p> <p><code>--cname name</code></p> <p>Use the specified name as the common name (CN) in the server certificate subject. This is normally the fully qualified domain name used by clients to connect to the server.</p> <p><code>--altNames entries</code></p> <p>Use the specified list of entries (comma or space separated) as the <code>subjectAlternateNames</code> list in the server certificate. Simple names are interpreted as <code>dns</code> entries. Entries may begin with <code>"dns:"</code> or <code>"ip:"</code> to indicate the type (for example, <code>"ip:192.168.0.1"</code> or <code>"dns:myHost"</code>). If no entries are specified, then reverse DNS is used to look up the registered names of the host's IP addresses.</p>
<code>list [--agent --server --index [--verbose]</code>	<p>Display certificate information for agent and/or server keystores or the CA certificate index. If no options are specified, both the agent and server keystores are listed.</p> <p><code>--agent</code></p> <p>List the contents of the agent keystore.</p> <p><code>--server</code></p> <p>List the contents of the server keystore.</p> <p><code>--index</code></p> <p>List the contents of the CA issued certificates index.</p> <p><code>--verbose</code></p> <p>Display additional details.</p>
<code>refreshCRL</code>	Refresh the certificate revocation list from the ElectricFlow server.
<code>revoke index</code>	Revoke a previously issued certificate by index.
<code>signCertificate csr crt</code>	Sign the certificate signing request provided in file <code>csr</code> and write the signed result to the file <code>crt</code> . The request is rejected by the CA if there is a matching certificate already in the CA database.
<code>updateAgentCertificate crt</code>	Install a previously signed certificate <code>crt</code> into the agent's keystore.

Server Communication Options

<code>--server host</code>	Address of the ElectricFlow server. Defaults to the value of the <code>COMMANDER_SERVER</code> environment variable. If that does not exist, it defaults to <code>localhost</code> .
<code>--securePort port</code>	HTTPS listener port on the server. Defaults to 8443.

Global Options

<code>--help</code>	Print the Help message.
<code>--version</code>	Print the version message.

Examples

Example 1: Configure an agent to talk to any server (untrusted mode)

This example generates a new self-signed certificate for the agent and recreates the keystore with no trusted authorities.

```
$ eccert initAgent -force
```

```
Generating keys
```

```
Generating certificate request
```

```
  cname=myAgent.company.com
```

```
  san=dns:myAgent.company.com
```

Example 2: Configure an agent to accept connections only from a single remote ElectricFlow server

This example generates a new certificate for the agent that is signed by the remote server's certificate authority and installs the signed certificate and its associated trust chain in the agent's keystore. After this point, the agent will only accept requests from the specified server and will be used as a trusted resource by the server.

```
$ ectool --server myserver login admin pw
```

```
$ eccert --server myserver initAgent -remote
```

```
Generating certificate request
```

```
  cname=myAgent.company.com
```

```
  san=dns:myAgent.company.com
```

```
Asking server 'myserver' to sign certificate
```

```
Importing 'CA:myserver.company.com' certificate
```

```
Importing 'jetty' certificate
```

Example 3: Configure an ElectricFlow server with additional host names in the certificate

This example regenerates the ElectricFlow Server Certificate, the specified common name, and alternate subject names to allow trusted connections with multiple external dns names.

```
$ eccert initServer --force --cname "myServer.company.com" --altNames "myServer,server2.company.com"
```

```
Generating keys
```

```
Generating certificate request
```

```
    cname=myserver.company.com
```

```
    san=dns:myserver,dns:server2.company.com
```

```
Signing server certificate
```

```
Importing 'CA:myserver.company.com' certificate
```

```
Importing 'jetty' certificate
```

ecconfigure

A command-line tool that can change configuration values for any locally installed ElectricFlow server, web, agent, or repository service. `ecconfigure` is a more user-friendly mechanism for configuring aspects of ElectricFlow that would otherwise require manual configuration file updates. `ecconfigure` actually manipulates relevant service configuration files on your behalf.

Usage

```
ecconfigure [options]
```

ElectricFlow agent configuration options

<code>--agentInitMemory=percent</code>	Initial java heap size as a percentage of the total system memory.
<code>--agentKeystorePassword=password</code>	Password used to access the agent's keystore..
<code>--agentMaxMemory=percent</code>	Maximum java heap size as a percentage of the total system memory.
<code>--agentInitMemoryMB=size</code>	Initial java heap size in MB.
<code>--agentMaxMemoryMB=size</code>	Maximum java heap size in MB.
<code>--agentLogFile=path</code>	Path where the agent log file should be written.

<code>--agentLogMaxFiles=max</code>	Maximum number of log files to accrue.
<code>--agentLogMaxSize=max</code>	Maximum size of each log file. The value can be suffixed with a unit (MB, KB, B). Without a unit, the value is interpreted as bytes.
<code>--agentPluginsDirectory=path</code>	The path used by the agent to get to the plugins directory on the ElectricFlow server where its resource definition lies.
<code>--agentLocalPort=port</code>	Port used by the ElectricFlow agent for HTTP communication on the <code>localhost</code> network interface.
<code>--agentPort=port</code>	Port used by the ElectricFlow agent for HTTPS communication on any network interface.
<code>--agentProtocol</code>	Protocol used by the agent.
<code>--agentProxyHost=host</code>	The IP address of the proxy server.
<code>--agentProxyPort=port</code>	The port of the proxy server.
<code>--agentNoProxyHosts=hosts</code>	Comma delimited list of hosts that should be reached directly, bypassing the proxy server.
<code>--agentEnableProxySettings=<0 1></code>	Enable or disable the proxy server configuration. If enabling for the first time, <code>--agentProxyHost</code> and <code>--agentProxyPort</code> must be specified.

Apache web server configuration options

<code>--webHostName=host</code>	The host name of the current machine in the form that users will typically use in their browser to access the web server.
<code>--webHttpPort=port</code>	The HTTP port of the web server.
<code>--webHttpsPort=port</code>	The HTTPS port of the web server.
<code>--webTargetHostName=host</code>	The host name of the ElectricFlow server to which the web server points.
<code>--webTargetHttpPort=port</code>	The HTTP port of the ElectricFlow server to which the web server points.
<code>--webTargetHttpsPort=port</code>	The HTTPS port of the ElectricFlow server to which the web server points.
<code>--webTimeZone=timezone</code>	The Olson TimeZone format (example: America/Los Angeles) for the php web server.

<code>--webPluginsDirectory=path</code>	The path used by the web server to get to the plugins directory on the ElectricFlow server to which it points.
<code>--webProxyUrl=url</code>	The IP address and port of the proxy server in the following format: <code>http://<IP_ADDRESS_PROXY>:<PROXY_PORT></code>
<code>--webNoProxyHosts=hosts</code>	Comma-delimited host list that should be reached directly, bypassing the proxy server.
<code>--webEnableProxySettings=<0 1></code>	Enable or disable the proxy server configuration. If enabling for the first time, <code>--webProxyUrl</code> must be specified.
<code>--webDLC=url</code>	The URL to use for downloadable content requests.

ElectricFlow Server configuration options

<code>--serverFileTransferPort=port</code>	The file transfer port of the server.
<code>--serverHttpPort=port</code>	The HTTP port of the server.
<code>--serverHttpsPort=port</code>	The HTTPS port of the server.
<code>--serverInitMemory=percent</code>	Initial java heap size as a percentage of the total system memory.
<code>--serverMaxMemory=percent</code>	Maximum java heap size as a percentage of the total system memory.
<code>--serverInitMemoryMB=size</code>	Initial java heap size in MB.
<code>--serverMaxMemoryMB=size</code>	Maximum java heap size in MB.
<code>--serverPasskeyFile=path</code>	Path to the server's passkey file.
<code>--serverProxyHost=host</code>	The IP address for the proxy server.
<code>--serverProxyPort=port</code>	The port for the proxy server.
<code>--serverNoProxyHosts=hosts</code>	Comma-delimited host list that should be reached directly, bypassing the proxy server.
<code>--serverEnableProxySettings=<0 1></code>	Enable or disable the proxy server configuration. If enabling for the first time, <code>--serverProxyHost</code> and <code>--serverProxyPort</code> must be specified.
<code>--serverZooKeeperConnection<ZooKeeper_servers></code>	The location of the ZooKeeper servers.

Repository Server configuration options

<code>--repositoryPort</code>	The repository server port.
<code>--repositoryInitMemory=percent</code>	Initial java heap size as a percentage of the total system memory.
<code>--repositoryMaxMemory=percent</code>	Maximum java heap size as a percentage of the total system memory.
<code>--repositoryInitMemoryMB=size</code>	Initial java heap size in MB.
<code>--repositoryMaxMemoryMB=size</code>	Maximum java heap size in MB.
<code>- repositoryStorageDirectory=path</code>	Path to the repository backing store. The artifact repository will use this directory to store artifacts.
<code>-- repositoryTargetHostName=host</code>	The host name of the ElectricFlow server to which the repository server points.
<code>-- repositoryTargetHttpPort=port</code>	The HTTP port of the ElectricFlow server to which the repository server points.
<code>-- repositoryTargetHttpsPort=port</code>	The HTTPS port of the ElectricFlow server to which the repository server points.
<code>-- repositoryTargetProtocol=<http https></code>	The protocol the repository server uses to talk to the ElectricFlow server.
<code>--repositoryProtocol</code>	The protocol the repository server uses to talk to client applications.
<code>--repositoryProxyHost=host</code>	The IP address for the proxy server.
<code>--repositoryProxyPort=port</code>	The port for the proxy server.
<code>--repositoryNoProxyHosts=hosts</code>	Comma-delimited host list that should be reached directly, bypassing the proxy server.
<code>-- repositoryEnableProxySettings= <0 1></code>	Enable or disable the proxy server configuration. If enabling for the first time, <code>--repositoryProxyHost</code> and <code>--repositoryProxyPort</code> must be specified.
<code>-- repositoryValidateFromDisk=<0 1></code>	Enable or disable disk validation.

General options

<code>-v, --version</code>	Displays the version information.
<code>-h, --help</code>	Displays this information.

Examples

Setting initial and maximum memory settings

For example, to set the ElectricFlow Server initial memory percentage to 21% and the maximum memory percentage to 31%, specify:

```
ecconfigure --serverInitMemory 21 --serverMaxMemory 31
```

If your ElectricFlow server, web server, or repository server is deployed behind a proxy server that inhibits certain Internet access, you can use `ecconfigure` to set proxy settings for each server in your installation.

To use the following perl scripts, remove the brackets ("`<`" "`>`"), and replace the bracketed example text with the values you need.

- To set ElectricFlow Server proxy settings:

```
ecconfigure --serverProxyHost <IP_ADDRESS_PROXY> --serverProxyPort <PORT>
--serverNoProxyHosts "<HOST1,HOST2>"
```

- To set Repository Server proxy settings:

```
ecconfigure --repositoryProxyHost <IP_ADDRESS_PROXY> --repositoryProxyPort
<PORT>
--repositoryNoProxyHosts "<HOST1,HOST2>"
```

- To set Web Server proxy settings:

```
ecconfigure --webProxyUrl <http://IP_ADDRESS:PORT> --webNoProxyHosts
<HOST1,HOST2,HOST3>
```

Changing the Apache web server port

Run `ecconfigure` with `--webHttpPort` and `--webHttpsPort`.

Your web server port setting will be changed appropriately in `httpd.conf`, `ssl.conf`, and `config.php`.

Configuring the backingstore location for the Artifact Repository

```
ecconfigure --repositoryStorageDirectory <new-path>
```

ecdaemon

`ecdaemon` is a "wrapper" program that can be used to start another program from an ElectricFlow job step—the "started" program will run as a daemon process. The ElectricFlow agent uses the facilities of the underlying operating system to make sure the process runs in a separate process group on a UNIX-based system, or outside of the normal "Windows Job" grouping in a Windows system. In either case, the ElectricFlow agent does not treat the process as one it should wait for or one it should try to "kill" if ElectricFlow needs to abort the step.

Use Cases

- `ecdaemon` is useful in the case where you are trying to deploy a "server-style" program in an ElectricFlow step. You do not want ElectricFlow to wait for that step to complete because it may run continuously, but you do want ElectricFlow to start the server program and then continue on to the next step.
- `ecdaemon` is useful if you want to "pre-load" some type of background process

`ecdaemon` launches the command and exits. Optionally, it sets a property in ElectricFlow with the `pid` of the program it spawned to make it possible for a later step to "kill" the daemonized program if desired.

For example:

```
ecdaemon c:/install.exe a b c
```

Using `ecdaemon`

```
$ ecdaemon
Usage: ecdaemon [options] cmd cmdArg1 cmdArg2 ...
       ecdaemon --version
```

Required:

command	The command to daemonize, followed by its args, if any.
---------	---

Options:

--pidProperty=prop-path	A property to set with the pid of the daemonized process.
--version	Print the version of this program and exit.

Command-line parsing

`ecdaemon` supports the standard UNIX-style `'--'` flag to indicate there are no more `ecdaemon` options and all subsequent options should be treated as simple arguments to the command.

This is particularly important for commands that themselves take `'--'` arguments.

For example:

```
ecdaemon /usr/bin/myserver --config /etc/myserver.conf
```

will not run properly because `ecdaemon` will attempt to parse the `--config` option instead of passing it to the `myserver` program.

The correct way to invoke `ecdaemon` in this case is:

```
ecdaemon -- /usr/bin/myserver --config /etc/myserver.conf
```

If you want to store the daemonized process's `pid` in a property, do so as follows:

```
ecdaemon --pidProperty /myJob/serverPid -- /usr/bin/myserver --config
/etc/myserver.conf
```

As a daemon process, any output goes to `/dev/null`, therefore no output file is generated.

ec-perl considerations

Use the `perl system ()` call to start `ecdaemon`.

`System ()` returns an exit status, "backticks" capture and return output that waits for the daemonized command to complete on Windows, and `exec` never returns at all if it is successful.

ecproxy

A driver script with built-in support for SSH. Every major operation can be overridden by defining a Perl function in the Proxy Customization field on the New Proxy Resource panel, available from the Resources page (by specifying which operation this function re-implements. These operations must have certain "signatures" for the driver to invoke them properly—the operations are listed and described below. For more detail, see the SSH implementation in `ecproxy.pl`.

ecproxy Algorithm

`ecproxy` invokes the operations detailed below to perform the following actions:

1. Uploads the command-file to a "workingDirectory" on the proxy target, using the protocol specified in the proxy config. Currently, only SSH is supported.
2. Creates a wrapper `sh` shell script that CD's to `workingDirectory`, sets "COMMANDER_" environment variables" that exist in the proxy agent's environment, and runs the command-file previously uploaded.
3. Uploads the wrapper script to `workingDirectory` on the proxy target.
4. Runs the wrapper script on the proxy target and streams its output to the proxy agent's `stdout`.
5. Deletes the local wrapper `shell` script, the remote wrapper, and remote command-file.
6. Exits using the exit code of running the wrapper script.

ecproxy Operations

getDefaultWorkingDirectory

Description	Computes the default working directory where a command needs to run on the proxy target, if the step is not defined with a working directory.
Arguments	None

Returns	The path to a directory as it would be accessed on the proxy target.
SSH implementation function	Not SSH-specific, so the function is 'getDefaultWorkingDirectory'
Existing implementation	Return <code>\$ENV{COMMANDER_WORKSPACE_UNIX}</code> ;
Reason to override	<p>If the "Working Directory" field is empty on a step that is going to run on a proxy target, the working directory for the step should be the workspace, just as it would be if the step were running on a non-proxy ElectricFlow agent.</p> <p><code>ecproxy</code> is guaranteed to run in the workspace directory on the proxy agent, but it is not guaranteed that the proxy target has the same path to the workspace.</p> <p>For example, the workspace on a Windows proxy agent is not the same as the path a Unix proxy target uses to access the workspace—so the existing implementation of this operation simply returns the UNIX path to the workspace. However, if the proxy target has a different path for accessing the workspace, the existing implementation will give the wrong answer. Thus, a user can provide a different implementation that gives the right answer.</p>

Note: This operation is applicable only if "Working Directory" is empty, and is used as the working directory on the proxy target in that case for running the command.

getDefaultTargetPort

Description	Computes the default port where the proxy target is listening for this protocol.
Arguments	None
Returns	The default port.
SSH implementation function	<code>ssh_getDefaultTargetPort</code>

Note: This operation is applicable only if the resource definition specifies no port value.

connect

Description	Opens a connection to the proxy target using the desired protocol.
Arguments	<code>host, port</code> (optional)
Returns	"connection-context hash-ref" on successful connection. This context can contain anything other functions can use to perform their tasks (for example, a connection handle).
Example 1	<pre>my \$context = connect('myhost', 22)</pre>
Example 2	<pre>my \$context = connect('myhost')</pre>

SSH implementation function `ssh_connect`

Note: Because the port is optional, the implementation of `connect` can default to whatever is reasonable for the protocol. This means the 'Proxy Target Port' need not be specified in the ElectricFlow web UI for proxied agents reachable on the default port.

uploadFile

Description	Uploads the given <code>srcFile</code> to the proxy target as <code>tgtFile</code> .
Arguments	<code>context</code> , <code>srcFile</code> (typically simple file-name), <code>tgtFile</code> (typically <code>workingDirectory/file-name</code>)
Returns	Nothing; on failure it does a 'die' with an appropriate error message.
Example	<pre>uploadFile(\$context, 'agent123.tmp', '/opt/work/joe/agent123.tmp')</pre>
SSH implementation function	<code>ssh_uploadFile</code>

generateWrapperScript

Description	Generates the script body that will run the command-file on the proxy-target in the <code>workingDirectory</code> .
Arguments	<code>workingDirectory</code> , <code>cmd</code> , <code>cmdArg1</code> , ..., <code>cmdFileName</code> (just the base-name, no directory)
Returns	A string containing the script to execute on the proxy target.
Example	<pre>generateWrapperScript('/opt/work/joe', 'perl', 'agent123.tmp')</pre>
SSH implementation function	Not SSH-specific, so the function is <code>'generateWrapperScript'</code>
Existing implementation	<code>cd workingDirectory</code> ; set <code>COMMANDER_</code> environment variables; run command-file, properly quoting the command and args.
Reason to override	If the proxy target does not have <code>sh</code> , the wrapper script needs to be written in a language available on the target.

uploadWrapperScript

Description	Uploads the wrapper-script code to the proxy target.
Arguments	<code>context</code> <code>workingDirectory.wrapperScriptBody</code>

Returns	The path to the wrapper-script on the proxy target.
Example	<pre>my \$wrapperFile = uploadWrapperScript(\$context, '/opt/work/joe', 'cd /opt/work/joe; perl agent123.tmp')</pre>
SSH implementation function	3.1, 3.1.1: <code>ssh_uploadWrapperScript</code>
3.1.2 and later	Not SSH-specific anymore, so the function is <code>'uploadWrapperScript'</code>

Notes:

1. This function must generate a uniquely named file that will not conflict with other `ecproxy` invocations that might be occurring in parallel steps. The recommended approach is to generate a file name containing the job-step-id.
2. Depending on the protocol and facilities available in the Perl implementation, you may or may not need to create a local `tmp` file to upload to the proxy target. If you do, record that fact in the context and clean up the local file in the `cleanup` operation. Setting the local wrapper file path in `$context->{wrapperFile}` is recommended because the default `cleanup` operation implementation looks for that string.

generateWrapperInvocationCommand

Description	Generates the command-line for running the wrapper script on the proxy target.
Arguments	<code>remoteWrapperFile</code> (path on proxy target)
Returns	A string containing the command-line for running the wrapper script file on the proxy target.
Example	<pre>my \$wrapperCmdLine = generateWrapperInvocationCommand (\$wrapperFile)</pre>
SSH implementation function	Not SSH-specific, so the function is <code>'generateWrapperInvocationCommand'</code>
Existing implementation	<code>Return "sh \$remoteWrapperFile";</code>
Reason to override	The default implementation of this function returns something like <code>'sh \$wrapperFile'</code> . If the wrapper script is not an <code>sh</code> script, or if you want to pass different arguments to the shell, you must override this function.

runCommand

Description	Runs the given command-line on the proxy target.
Arguments:	<code>context, cmdLine</code>

Returns	exit-code from running the command on the proxy target, <code>undef</code> if the command could not be run for some reason.
Example	<code>runCommand(\$context, \$wrapperCmdLine)</code>
SSH implementation function	<code>ssh_runCommand</code>

cleanup

Description	Performs any cleanup task after the command has completed on the proxy target. Typically, it deletes any locally created temp files and uploaded files on the proxy target.
Arguments	<code>context</code> , <code>cmdFile</code> , <code>wrapperFile</code> (both are of the form <code>workingDirectory/file-name</code>)
Returns	1 on success, <code>undef</code> on failure.
Example	<code>cleanupTarget(\$context, '/opt/work/joe/agent123.tmp', '/opt/work/joe/cmdwrapper.123.tmp')</code>
SSH implementation function	3.1, 3.1.1: <code>ssh_cleanup</code>
3.1.2 and later	Not SSH-specific anymore, so the function is <code>cleanup</code>

Note: The default implementation deletes the locally created wrapper script file whose path is stored in `$context->{wrapperFile}`, if it exists. Thus, if the `uploadWrapperScript` operation is overridden, it is recommended the overriding function set this attribute—that way, `cleanup` need not be overridden.

ping

Description	A test to see if the proxy target is usable.
Arguments	<code>host</code> , <code>port</code> (optional)
Returns	1 on success, <code>undef</code> on failure.
Example	<code>ping('myhost', 22)</code>
SSH implementation function	Not SSH-specific, so the function is <code>ping</code> .
Existing implementation	Opens a socket connection to the proxy target on the desired port.
Reason to override	The existing implementation may be deemed too simple for doing a ping; overriding <code>ping</code> to open a connection and do some protocol-specific handshaking might be more appropriate for some protocols / use cases.

Available Helper Functions

To make proxy customization easier, `ecproxy` provides the following helper functions.

mesg

Description	Debug logging function. Writes to the file referenced in the <code>ECPROXY_DEBUGFILE</code> environment variable (if it exists). No-op otherwise.
Arguments	<code>message</code>
Example	<pre>mesg("myCleanup: about to delete \$cmdFile on proxy target");</pre>

Note: This function automatically adds a newline to whatever it emits, so the caller does not have to incorporate a newline in message.

readFile

Description	Reads a file.
Arguments	<code>fileName</code>
Returns	Contents of the file. If there is an error, it returns an empty string.
Example	<pre>my \$data = readFile("foo.txt");</pre>

writeFile

Description	Creates a local file containing data.
Arguments	<code>fileName, data</code>
Returns	<code>1</code> on success, <code>undef</code> on failure.
Example	<pre>writeFile("myWrapper.\${ENV{COMMANDER_JOBSTEPID}}.cmd", "perl foo.pl")</pre>

initDispatcher

Description	Initialize the operation dispatcher map to point to functions for the given protocol. For each operation, <code>initDispatcher</code> checks if a function named <code>protocol_operation</code> exists, and if so, assigns that function as the implementation for that operation.
Arguments	<code>protocol</code>

Example `initDispatcher("ssh")` sets the "connect" operation to run "ssh_connect", "uploadFile" => "ssh_uploadFile", etc.

setOperation

Description Sets the implementation of an operation to be a particular function.

Arguments operation,function. The 'function' argument may be the name of a function or a reference to a function.

Example `setOperation("ping", "my_ping");` sets the "ping" operation to run the "my_ping" function

Example `setOperation("ping", \&my_ping);` same as above, but using a function ref

Note: This function manipulates the `gDispatcher` hash, but provides a safe interface to it.

loadFile

Description Load proxy customizations from a file.

Arguments fileName

Example `loadFile("custom.pl")`

setSSHKeyFiles

Description Set the paths to the public and private key files that ssh will use to authenticate with the proxy target.

Arguments publicKeyFile,privateKeyFile

Example `setSSHKeyFiles('c:\foo\pub.key', 'c:\foo\priv.key')`

Note: This is very useful on Windows proxies, where there is no reasonable default for ssh to use.

setSSHUser

Description Set the name of the user to authenticate with the proxy target.

Arguments userName

Example

```
setSSHUser('user1')
```

Note: By default, the user name the agent is "running as" is used to log into the proxy target. If key-based authentication is configured on the target system such that 'agentUser' can log into the 'user1' account on the proxy target, this function leverages that configuration.

useMultipleSSHSessions

Description

Normally, ecproxy uses one ssh session with a number of "channels" to perform tasks like uploading files, running the command, and running a cleanup command on the proxy target. Some SSH servers don't allow this. This method configures ecproxy to use a separate SSH session for each operation; this requires authenticating with the SSH daemon on the proxy target several times, and thus it may perform worse than the single-session-multi-channel mode.

Arguments

None

Example

```
useMultipleSSHSessions()
```

Examples

Specify public/private key files for SSH

1. Set the `proxyCustomization` property on the resource like this: `?setSSHKeyFiles ('c:\foo\pub.key', 'c:\foo\priv.key');`
2. Set the `ECPROXY_SSH_PRIVKEYFILE` and `ECPROXY_SSH_PUBKEYFILE` environment variables on the proxy agent as system environment variables.

Override one of the operations (for example, to enable SSH connection with username/password)

Set the `proxyCustomization` property on the resource like this: `?sub myConnect($$) {...}`
`setOperation("connect", \&myConnect);`

Load proxy customizations from a file rather than having all the logic in the `proxyCustomization` property on the resource

Set the `proxyCustomization` property on the resource like this: `?loadFile('c:\foo\custom.pl');`

Implement a whole new protocol

Specify protocol as 'myproto' and have a proxy customization block like this:

```

sub myproto_getDefaultTargetPort() {
  ...
}
sub myproto_connect($;$) {
  ...
}
sub myproto_uploadFile($$$) {
  ...
}
sub myproto_uploadWrapperScript($$$) {
  # Note: As of 3.1.2, the default implementation is likely good enough, so
  it may not be necessary to define this override
  ....
} sub myproto_runCommand($$) {
  ...
} sub myproto_cleanup($$$) {
  # Note: As of 3.1.2, the default implementation is likely good enough, so
  it may not be necessary to define this override
  ....
}
# Initialize the dispatcher to run these functions
initDispatcher("myproto");

```

Override ping to do a connect operation (which does a full protocol handshake, authentication, and so on)

Write a specialized ping function for the proxy customization like this:

```

sub heavy_ping($$) {
  my ($host, $port) = @_;
  return ssh_connect($host, $port);
}
setOperation("ping", \&heavy_ping);

```

"Real World" Examples

ClusterExec

A basic integration for using `clusterupload` and `clusterexec` to reach a proxy target is here. It has been tested on a Windows target with a Cygwin installation. It will not work "out-of-the-box" because it makes the following assumptions:

- The proxy target has `sh` and other UNIX tools (for example, `rm`).
- The locations of the `clusterexec` and `clusterupload` binaries are hard-coded at the top of the proxy customization.

To make this proxy customization work on a Windows machine that does not have Cygwin, the `generateWrapperScript` operation would need to be overridden with a function that generates a `cmd` batch script, and the `generateWrapperInvocationCommand` operation would have to be overridden to generate a `"cmd /c ..."` command rather than `"sh ..."`.

MySQL

The idea here is that the proxy target need not be a host for running arbitrary commands. It could be a special entity like a db. This integration uses the `mysql clt` to run the step command (which should be SQL) on the db referenced by the proxy target host and port.

A bare-bones integration with MySQL:

```
# Set the path to the mysql binary; if the directory is in the proxy agent's
# PATH, this variable can simply contain the name of the executable.

my $gMySQL = "c:/cygwin/usr/local/tools/i686_win32/bin/mysql.exe";

sub mysql_getDefaultTargetPort() {
    return 3306;
}

sub mysql_connect($;$) {
    # This "protocol" implementation is just going to use the mysql
    # command-line tool, so just save off host/port.

    my $host = $_[0];
    my $port = $_[1] || mysql_getDefaultTargetPort();

    return {host => $host,
            port => $port};
}

sub mysql_uploadFile($$$) {
    my ($context, $cmdFile, $rmtCmdFile) = @_;

    # We do not need to upload the command-file to the proxy target.
    # We are going to run the mysql clt on the proxy agent to run
    # the query (contained in the local command-file),
    # so just save off the name of the command-file.

    $context->{cmdFile} = $cmdFile;
}

sub mysql_uploadWrapperScript($$$) {
    my ($context, $workingDir, $wrapperScript) = @_;

    # This has no meaning for this integration. No-op.
}

sub mysql_runCommand($$) {
    my ($context, $cmdLine) = @_;

    # cmdLine is a command-line for running the wrapper script, which
    # has no meaning for this integration. We just want to run
    # 'mysql' for the desired host/port with the command-file.

    system("$gMySQL -D commander -h $context->{host} -P $context->{port} " .
          "-u commander -pcommander -e \"source $context->{cmdFile}\"");
}

sub mysql_cleanup($$$) {
    # We didn't create any temp files. No-op.
}

# Initialize the dispatcher to run these functions
initDispatcher("mysql");
```

Android

This example uses the `adb` tool to upload files to the device and run commands on it. Initial testing has been only against the android emulator, but it is implemented in such a way that it should work against a real android device attached using USB to the proxy agent, or a device on the network.

A first attempt at proxying to android devices:

```
# Set the path to the adb binary; if the directory is in the proxy agent's
# PATH, this variable can simply contain the name of the executable.

my $gADB = "c:/android-sdk-windows-1.6_r1/tools/adb.exe";

sub android_getDefaultTargetPort() {
    # Not sure what a good meaningful value is here.
    return 0;
}

sub android_connect($;$) {
    # This "protocol" implementation uses the adb
    # command-line tool. Depending on the value of
    # host, construct the appropriate adb command-line
    # argument.

    my $host = $_[0];
    my $context = {};
    # if ($host eq "emulator") {
    if ($host eq "localhost") {
        # We want to talk to the emulator running on this host.
        $context->{targetArg} = "-e";
    } elsif ($host eq "usb") {
        # We want to talk to the single android device connected
        # to the computer via a USB.
        $context->{targetArg} = "-d";
    } else {
        # This must be the serial number of some device somewhere.
        $context->{targetArg} = "-s $host";
    }
    return $context;
}

sub android_uploadFile($$$) {
    my ($context, $srcFile, $tgtFile) = @_;
    my($filename, $directories) = fileparse($tgtFile);

    my $result = `$gADB $context->{targetArg} push $srcFile
"/data/tmp/$filename" 2>&1`;
    if ($? != 0) {
        die ("android_uploadFile: Error uploading file $srcFile to
/data/tmp/$filename: $result\n");
    }
}

sub android_runCommand($$) {
    my ($context, $cmdLine) = @_;

    # cmdLine is a command-line for running the wrapper script, which
```

```
# has no meaning for this integration. We just want to run
# 'adb' for the desired device with the command-file.

system("$gADB $context->{targetArg} shell $cmdLine");
}
sub android_cleanup($$$) {
  my ($context, $remoteCmdFile, $remoteWrapperFile) = @_;

  # This was copied from ssh_cleanup except that we do "rm",
  # not "rf -f".

  mesg("cleaning up");

  # Delete the locally generate wrapper file.
  unlink($context->{"wrapperFile"});

  # Delete the cmd-file and wrapper script file on the proxy target.
  $gDispatcher{"runCommand"}($context,
    "rm $remoteWrapperFile $remoteCmdFile");
}
sub android_ping($;$) {
  my ($host, $port) = @_;
  $port = $gDispatcher{"getDefaultTargetPort"}() unless isPortValid($port);

  my $socket = IO::Socket::INET->new(PeerAddr => $host,
                                     PeerPort => $port,
                                     Proto    => "tcp",
                                     Type     => SOCK_STREAM)
    or die "Couldn't connect to $host:$port : $@\n";
}

# Initialize the dispatcher to run these functions
initDispatcher("android");
1;
```

ecremotefilecopy

When ElectricFlow agents (on platforms other than Linux or Windows) run steps that create log files in a workspace the ElectricFlow web server cannot access (through Linux or Windows agents), use *ecremotefilecopy* to recreate job logs so they are visible on those ElectricFlow agents, which then enables the web server to retrieve and render those log files.

Using `postp` and *ecremotefilecopy*, the log file is populated and recreated in a workspace accessible to the ElectricFlow web server, allowing the Job Details page to display the log file. Although this functionality is supported, it is not a recommended method of operation. This method should be used only as a last resort when a shared file system (between alternate agents and primary platform agents [Linux and Windows]) is not possible.

The reasons *ecremotefilecopy* is not recommended are:

- You will not see logs in real time. Logs are not visible until the "recreate step" has completed running.
- There is a performance penalty, especially when running with large files.

Setting up the process

- Create a "Setup" step in your procedure
- Update the Postprocessor field for each step whose results you want to see on the server.
- Add a step (one or more times) to the procedure to recreate the ElectricFlow server log files.

Creating a Setup Step

In your procedure, create a step called "Setup". This step needs to be in your procedure **before** any step running on a remote workspace.

Note: This is your top-level procedure, not a subprocedure.

- Navigate to your procedure.
- To create a new step, click the Command step link.
- Set the fields as follows:

Step Name: Setup

Command(s): `ecremotefilecopy setup`

Resource: local

Workspace: you have two choices:

- use default
- use: `alternateWorkspaceForDisplay`

There is a property on the Workspace called `alternateWorkspaceForDisplay`, which is a secondary location to look for workspace files. This secondary location is used when the workspace files are not accessible to the web server. If the Apache server cannot locate the file in the original workspace, it looks in the alternate one.

Update the Postprocessor field for steps in your procedure

This step defines a postprocessor that will run at the end of the steps you specify. Add the following information to every step running on a remote agent if you want to see its results in the web interface.

- Navigate to a procedure and a step.
- In the Postprocessor field, enter:
`postp --check none --loadProperty /myJob/jobSteps[Setup]/postpExtensions`
- If you are using `postp` in this step to scan your step log for errors, warnings, and so on also, omit "`--check none`" from the invocation line.

Add a Final Step to your procedure

This step adds a new step at the end of your existing procedure. This step finds all properties created by the postprocessor, then reads the properties and creates local log files based on the properties, then deletes the properties.

- Navigate to your procedure.
- To create a new step, click the Command step link.
- Set the fields as follows:

Step Name: `Recreate the Log Files`

Always run step: (Check the box)

Command(s): `ecremotefilecopy recreateFiles`

Resource: `local`

Workspace: `default`

After the final step runs, you should see links (icons) displayed in the Log column on the Jobs Details page.

Click the icon to display the log file.

Copying Other Files from the Workspace

By default, `ecremotefilecopy` copies only `postp` log and `diag` files, and step logs. You can also copy other files from the workspace using a function named `postpEndHook2`.

You must do the following in your step:

1. Make sure that the file you want to copy is in the step workspace. (It can be copied there, created there, etc.)
2. For your procedure, create a property (named `postpEndHook2`, for example).
3. Define a function named `postpEndHook2` inside the property. For example:

```
sub postpEndHook2() {  
  
    # Missing param does not cause an error  
    $::gCommander->abortOnError(0);  
  
    # Add filename to a "special" property such that it will be picked up by ecr  
    emotefilecopy  
    my $fileName = 'paul.txt';  
    copyFileToProperty ($fileName);  
  
    # Restore default error handling  
    $::gCommander->abortOnError(1);  
}
```

4. Add the following line in the Postprocessor field of this step:

```
postp --check none --loadProperty /myJob/jobSteps[Setup]/postpExtensions --loadP  
roperty /myProcedure/postpEndHook2
```


ZKConfigTool

You can set ZooKeeper server with configuration information that all ElectricFlow server nodes will use in a clustered configuration. This command-line tool that imports your ElectricFlow database configuration information into your ZooKeeper server. The following minimum set of files is imported:

- database.properties
- keystore
- passkey
- commander.properties

Prerequisites

- The ElectricFlow Tools package must be installed on the system.
- The system must be running a version of Java supported by ElectricFlow. Java is automatically installed on a system with the ElectricFlow software as part of the Tools install.
- The ZooKeeper software must be installed on the network.

Command

```
$ java -jar zk-config-tool-jar-with-dependencies.jar -<options>
```

Usage: ZKConfigTool

Option	Description
-commanderPropertiesFile <path_to_file>	Import the ElectricFlow commander.properties file.
-databasePropertiesFile <path_to_file>	Import the ElectricFlow database.properties file.
-help	Show the command Help.
-keystoreFile <path_to_file>	Import the ElectricFlow keystore file.
-passkeyFile <path_to_file>	Import the ElectricFlow passkey file.
-wrapperConfFile <path_to_file>	Import the wrapper.conf file.
-writeFile <path_on_zookeeper> <path_to_file>	Write the specified file to the ZooKeeper server.

Import Files

Run the `ClusterTool` command to completely populate the ZooKeeper server with configuration information. The system must have the Tools install and can be able to communicate with ZooKeeper.

- If you run the command with defined file options, the tool immediately attempts to contact the specified ZooKeeper system and import the specified files.

Example command:

```
COMMANDER_ZK_CONNECTION=10.168.33.10:2181 java -jar zk-config-tool-jar-with-  
dependencies.jar --databasePropertiesFile database.properties --keystoreFile  
keystore --passkeyFile passkey --wrapperConfFile wrapper.conf --  
commanderPropertiesFile commander.properties
```

- If you run the command without defined file options, the tool displays a series of prompts for the required ElectricFlow information. Once the correct information is entered, the tool contacts the specified ZooKeeper system to import the specified files.

Example command:

```
$ COMMANDER_ZK_CONNECTION=10.168.33.10:2181 java -jar zk-config-tool-jar-with-  
dependencies.jar
```

ClusterInfoTool

Use `ClusterInfoTool` to get information on the running ElectricFlow server cluster from ZooKeeper.

Prerequisites

- The ElectricFlow server cluster must be installed and running on the network.
- Configuration files that all ElectricFlow server nodes will use in a clustered configuration must be uploaded to Apache ZooKeeper server using the `ZKConfigTool`.
- The ZooKeeper cluster must be running an odd number of Zookeeper nodes and there must be a leader node.
- The system must be running a version of Java supported by ElectricFlow. Java is automatically installed on a system with the ElectricFlow software as part of the Tools installation.

Locations

The ElectricFlow installer adds the `ClusterInfoTool` to the following default locations:

- **Windows:** `C:\Program Files\Electric Cloud\ElectricCommander\server\bin\cluster-info-tool-jar-with-dependencies.jar`
- **Linux:** `/opt/electriccloud/electriccommander/server/bin/cluster-info-tool-jar-with-dependencies.jar`

Command

```
$ java -DCOMMANDER_ZK_CONNECTION=<ZooKeeper_Server1_IP>:2181,<ZooKeeper_Server2_IP>:21  
81,<ZooKeeper_Server3_IP>:2181 -jar cluster-info-tool-jar-with-dependencies.jar
```

where `DCOMMANDER_ZK_CONNECTION` must point to the ZooKeeper system to which the ElectricFlow server cluster is connected.

Output

This is sample output generated by `ClusterInfoTool`:

```
Checking /commander/jgroups/hornetq:  
582d3642-9736-e87f-4e19-c22d7776ccab ->
```

```

WIN-M3Q09A2PNFP-21066    6c80e9a9-2fed-6352-e437-fe76b65aa80d    10.0
.175.78:5446            F
WIN-M3Q09A2PNFP-28728    0afa39df-d072-b05a-034a-aed74b7a39ee    10.0
.238.179:5446            F
WIN-M3Q09A2PNFP-28295    c69f0430-df04-bf24-1294-b471a4a3f151    10.0
.2.207:5446              F
WIN-M3Q09A2PNFP-15869    582d3642-9736-e87f-4e19-c22d7776ccab    10.0
.2.206:5446              T

```

Checking /commander/jgroups/commander:

```
0c32103a-d4e7-da91-f8c4-1fd8e125c156 ->
```

```

WIN-M3Q09A2PNFP-19713    0c32103a-d4e7-da91-f8c4-1fd8e125c156    10.0
.2.206:5447              T
WIN-M3Q09A2PNFP-14432    75afe3eb-7a04-d160-c8d0-d64f8ac3c796    10.0
.175.78:5447            F
WIN-M3Q09A2PNFP-33530    1cf1795f-e658-3cae-c515-546082bfffec9    10.0
.238.179:5447            F
WIN-M3Q09A2PNFP-60743    cf8b4201-3fac-71c4-18da-0a885b3e1e61    10.0
.2.207:5447              F

```

How to interpret ClusterInfoTool output:

- The nodes /commander/jgroups/hornetq and /commander/jgroups/commander contain information on these JGroups clusters:
 - commander for the ElectricFlow server cluster
 - hornetq for the HornetQ cluster
- The children nodes under each of the JGroups nodes represent the participating ElectricFlow servers in the cluster. Each child node entry is in this form:

```
<Logical Name>    <UUID>    <IP address>:<port>    T|F
```

- The number of entries in both JGroups nodes should be the same, with matching IP addresses but with different port numbers and distinct logical names and UUIDs. The coordinator node in each JGroups cluster is identified with a 'T' against its entry.

Automation Platform Tasks

All the topics in this section are about a particular feature or function in the ElectricFlow automation platform. They are the same as the topics in the "Web Interface Help" section of the Help system on the automation platform UI.

From the automation platform UI, you can access the topics as follows:

- All ElectricFlow Help topics above this directory are expanded, user-guide style Help topics about a particular ElectricFlow feature or function. Direct access to these Help topics is from the Help system Table of Contents. Although these topics are not directly linked within the ElectricFlow product, many "page-specific Help topics contain links to one or more of these topics to provide more detailed information quickly.

- The Help topics listed below this directory are "page-specific" Help topics, which means each of these topics contains specific information for the automation platform web page you are viewing. To access these Help topics while using ElectricFlow, click the **Help** link at the top-right corner on any automation platform web page. (The topics in this section of the AllHelp directory are listed singularly or in groups as is appropriate.)

Access Control

Use this page to view and modify access privileges for a particular ElectricFlow object. Depending on the object where you want to set permissions, you will see that object's name as part of the page title (above the tables).

For example, if you clicked the Access Control link from a Project Details page, you will see the project name as part of the Access Control page title.

Reading and Using This Page

- This page displays one or more access control lists. The top list contains entries for the object itself (specified in the page title) and also identifies the object.
For example, if the heading for the top list reads "Privileges for Procedure: buildAndTestAll," you are viewing access privileges for a procedure named "buildAndTestAll". Click the object name to view the main page for that object.
- Typically, you will see more than one list on the page. Each list below the first one contains privileges for an object that "contains" the objects above it.
For example, a project contains all of its procedures and a procedure contains all of its steps. Privileges for the top-level object are determined by all the privileges in all of the displayed lists. The lists form an "inheritance chain" where it each object "inherits" permissions from the objects below it on the page.
- When a user attempts a particular operation on the object, ElectricFlow examines the lists on this page from top to bottom. If there is an entry specifying "deny" for the user (or a group containing the user) in the top list, access is denied. Otherwise, if an entry specifies "allow" for the user (or a group containing the user) in the top list, access is allowed. If access is neither allowed nor denied by the top list, ElectricFlow proceeds to the next list and processes it in the same way.
Note: If access is neither allowed nor denied by any list, ElectricFlow denies access.
- The inheritance mechanism makes it easy to control access for a large number of objects in a single place.
For example, project access control entries automatically apply to new objects created within the project. Each new object in the project has an empty access control list, but will inherit from the project.

Using the Links...

Use these links to add or increase Access Control for an object.

- **Add User** - Use this link to add permissions for a specific user.
- **Add Group** - Use this link to add permissions for a specific group, which means all users in that group would have the permissions allowed to the group.
- **Add Project** - Use this link to set or redefine permissions for a project.

- **Break Inheritance - Caution: Be very careful if you break inheritance!** If you use the "Break Inheritance" action for any list, no additional inheritance occurs *below* that list and you no longer see other lists on this page. This action is useful if you want privileges for an object to be totally different than its containing object. If an object has no entries in its access control list and you break inheritance for that object, you make the object *completely inaccessible* - you will not even have the "Change Permissions" privilege, so you cannot restore inheritance. If this occurs, see your system administrator to restore inheritance.
- Actions:
 - **Edit** - Use this link to modify current permissions, but be careful if you modify permissions in an *inherited* access control list. Modifying inherited access control affects all other objects that inherit from the same list.
 - **Delete** - Deletes the current privileges granted for that user, group, or project.

Privilege Definitions

The following four privilege types (for each ElectricFlow object) can be assigned *allow*, *deny*, or *inherit* permission.

Read - Allows object contents to be viewed.

Modify - Allows object contents (but not its permissions) to be changed.

Execute - If an object is a procedure or it contains procedures (for example, a project), this privilege allows object procedures to be invoked as part of a job. For resource objects, this privilege determines who can use this resource in job steps.

Change Permissions - Allows object permissions to be modified.

For more information, see the main [Access Control](#) Help topic. This Help topic also contains two examples that illustrate how you might use Access Control to increase ElectricFlow security.

Access Control - defining entries

Use this page to define access control entries selected from a filtered list of ElectricFlow users, groups, or projects, or by providing an explicit name.

Enter information in the fields as follows:

Note: The following instructions are generic. For example, if you chose Add User, the Filter field will be labeled "User Filter".

Field Name	Description
Radio selector	Choose between selecting "Search for existing..." or select "Provide explicit ... name". Providing an explicit name bypasses the search and allows you to enter an arbitrary name.

Field Name	Description
Filter	Enter a partial name to retrieve a list of all users/groups/projects that include the partial name you supplied.
Include Inactive	<p>[User/Group only] - Use this checkbox only if requesting external LDAP or Active Directory providers.</p> <p>If "checked," all available users or groups are returned.</p> <p>If not checked, only those users or groups known to the ElectricFlow server are returned, for example, users who have logged in or groups containing users who have logged in.</p>

Click **OK** to retrieve your information.

If you entered filter criteria, the page will refresh with a list of matches. Select any row in this table to create an access control entry for that principal.

Column descriptions

Name - The name of the user/group/project.

Location - Only viewable for users or groups from non-local directory providers. The name combined with the location is the unique identifier for this user or group, local to the ElectricFlow server.

Privileges - create new or edit existing privileges

Use this page to create or modify an access control list entry. Each entry names a *principal*, which can be either an individual user or a group, and defines four privileges for the principal.

For each of the four privileges, **Read**, **Modify**, **Execute**, or **Change** Permissions:

- Creating a new entry or modifying an existing entry allows you to select either Inherit, Allow, or Deny access entries.
- On the server access control table, "Don't Care" replaces the "Inherit" option.

For more information about access control, see the [Access Control](#) Help topic.

Artifacts

This page displays all artifacts available on this ElectricFlow server.

Links and actions at the top of the table

- **New Search** - Use this link to go to the Define Search page to filter the Artifacts page view.
- **Create Artifact** - Use this link to go to the New Artifact page to create a new artifact.
- The "star" icon allows you to save this page to your Home page for quick access.

Column descriptions

Column Name	Description / Actions
Name	The name of the artifact—a system-generated name created by combining the Group Id and Artifact Key components. Note: Selecting an artifact name takes you to the Artifact Details page for that artifact.
Group Id	The user-generated group name for this artifact. This field may consist of alphanumeric characters, spaces, underscores, hyphens, and periods.
Artifact Key	The "key" component of the artifact name.
Description	The plain text or HTML description previously supplied for this artifact.
Actions	Edit - Use this link to go to the Edit Artifact page to modify the artifact on this row. Delete - Use this link to delete the artifact on this row.

Artifact Details

This page displays detailed information about an artifact.




Links and actions at the top of the table

- **Edit** - Use this link to go to the Edit Artifact page to make any necessary changes for this artifact's description or artifact version name template.
- **Access Control** - Use this link to set privileges for this artifact. *For more information*, see the [Access Control](#) Help topic.
- The "star" icon allows you to save this job information to your Home page.

General Information section

This section displays the artifact name (Artifact), Group Id, Artifact Key, and any Description previously supplied for this artifact.

Artifact Details – exports:project

 Edit  Access Control 

General Information


Artifact: exports:project
Group Id: exports
Artifact Key: project
Description:

The "tabbed" sections

The next section of tabs allows you to select the type of information you want to see.

Artifact Versions table

This table displays all artifact versions included in this artifact.

Artifact Versions			Properties
Artifact Version	Description	Actions	
com.ec.myapp:1.0		 	

Column descriptions

Artifact Version - Click on an artifact version name to go to the Artifact Version Details page for that artifact version.

Description - The previously supplied description, if any, for this artifact version.

Actions

- Click the **Edit** link to go to the Edit Artifact Version page.
- Click the **Delete** link to delete this artifact version from the artifact.

Properties table

This table contains custom properties for this artifact.

- Click on a Property Name to edit that property.
- If a "folder icon" precedes a property name, it denotes a Nested Property Sheet.
- This section also provides **Create Property**, **Create Nested Sheet**, and **Access Control** links to add more custom properties or change or set privileges on this artifact.

Artifact Versions			Properties
Property Name	Value	Description	Actions
Status	New		

Create Property | Create Nested Sheet | Access Control

Artifact - create new or edit existing artifact

To create a new artifact

Enter information in the fields as follows:

Field Name	Description
Group Id	Enter a group name of your choice for this artifact. A Group Id (groupId) acts as a namespace for grouping related artifacts. The idea is that artifact "sdk" for group "platform" can co-exist with artifact "sdk" in group "ui" without there being any name collisions. This field is limited to alphanumeric characters, spaces, underscores, hyphens, and periods.
Artifact Key	Enter an identifier of your choice for this artifact. This field is limited to alphanumeric characters, spaces, underscores, hyphens, and periods.
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code> .
Artifact Version Name Template	<p>A template for the names of artifact versions published to this artifact.</p> <p>For example:</p> <pre>\$/myArtifactVersion/groupId:\$/myArtifactVersion/artifactKey:\$/myArtifactVersion/version]</pre> <p>Produces a name like: platform:sdk:1.0.0-36</p> <p>Only absolute property paths and /myArtifactVersion-based paths are supported. If not specified, the "Artifact Version name template" server setting is used.</p>

Click **OK** after filling-in the fields and to go to the Artifacts page to see your new artifact listed in the first column.

To edit an artifact

- Click the **Access Control** link at the top of the page to add permissions to this artifact.
- You can add a Description or modify your existing description.
- You can add an Artifact Version Name Template or modify your existing template.
- Click **OK** to save your changes and return to the Artifacts page.

Artifact Versions

This page displays all artifact versions available on this ElectricFlow server.

Links and actions at the top of the table

- **New Search** - Use this link to go to the Define Search page to locate artifacts, repositories or other related information.
- The "star" icon allows you to save this page to your Home page for quick access.

Column descriptions

Column Name	Description / Actions
Name	The name of the artifact version. Select an artifact version name to go to the Artifact Version Details page for that artifact version.
Artifact	The name of the artifact. Select an artifact name to go to the Artifact Details page for that artifact.
Group Id	The name of the group where this artifact version is a member.
Artifact Key	The "key" identifier for this artifact version.
Version	The artifact version represented, by default, as <code>major.minor.patch-qualifier-buildNumber</code> . Note: You will not see all 5 artifact version components specified if only 3 components were defined.
State	The current state of this artifact version. Possible values are: <code>available publishing unavailable</code> .
Modify Time	The last time this artifact version was modified.
Actions	Edit - Use this link to go to the Edit Artifact Version page to modify the artifact version on this row. Delete - Use this link to delete the artifact version on this row.

Artifact Version Details

This page displays detailed information about an artifact version.

Links and actions at the top of the table

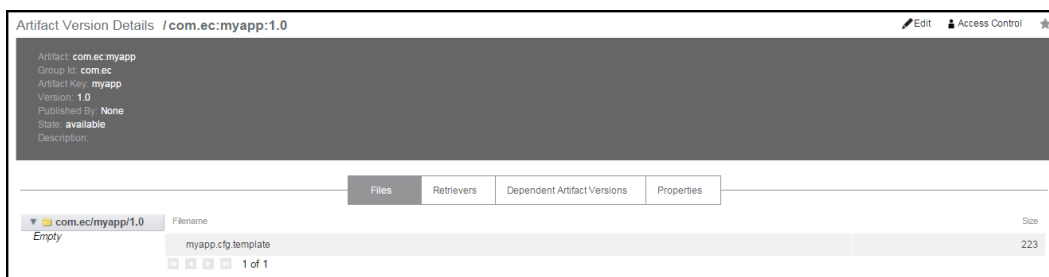
- **Edit** - Use this link to go to the Edit Artifact Version page.
- **Access Control** - Use this link to set privileges for this artifact version. *For more information*, see the [Access Control](#) Help topic.
- The "star" icon allows you to save this artifact version information to your Home page.

General Information section

This section lists current information about the artifact version. While most of the information is static, two items link to other pages:

Artifact - Click the artifact name to go to the Artifact Details page for the "owning" artifact.

Published By - Click this link to go to the Job Details page for the job that published this artifact version. If this artifact version was published by an end user outside of a job context, this field will display None.



The "tabbed" sections

The next section of tabs allows you to select the type of information you want to see.

Files

This section shows the collection of files included in this artifact version. The left-pane displays the artifact version directory structure and the right-pane displays files and subdirectories for the currently selected folder in the left-pane. This structure is similar to Windows Explorer.



Retrievers table

This table lists the 50 most recent jobs that retrieved this artifact version and includes links to the Job Details page for those jobs.

Files	Retrievers	Dependent Artifact Versions	Properties
-------	------------	-----------------------------	------------

Currently, there are no records to display in this list.

Dependent Artifact Versions table

This section displays dependent artifact version queries registered with the artifact version during a publish operation. These queries are evaluated during retrieval and the most current artifact version that matches each query is returned along with the primary artifact version.

Files	Retrievers	Dependent Artifact Versions	Properties
-------	------------	-----------------------------	------------

Currently, there are no records to display in this list.

Properties table

This table contains custom properties for this artifact version.

- Click on a Property Name to edit that property.
- If a "folder icon" precedes a property name, it denotes a Nested Property Sheet.
- This section also provides **Create Property**, **Create Nested Sheet**, and **Access Control** links to allow you to add more custom properties.

Files	Retrievers	Dependent Artifact Versions	Properties
-------	------------	-----------------------------	------------

Currently, there are no records to display in this list.

Create Property | Create Nested Sheet | Access Control

Artifact Version - edit an existing artifact version

This page displays an existing artifact version you can modify.

Links and actions at the top of the table

- **Access Control** - Use this link to add or change permissions for this artifact version.
- The "star" icon allows you to save this page to your Home page for quick access.

To edit an artifact version

Field Name	Description / Action
Name	You can type-over the artifact version name in this field to change or modify the existing name.
Description	You can add a description or modify your existing description.

Field Name	Description / Action
Publisher	This is the job that completed the publish operation. Click this link to go to the Job Details page for this job. If this artifact version was published outside of a job context (for example, from ectool), the value for this field is "None".
State	<p>An artifact version can be in one of three states: <code>available</code>, <code>unavailable</code>, and <code>publishing</code>.</p> <p>If the artifact version is:</p> <ul style="list-style-type: none"> • in the Publishing state, the artifact version cannot be retrieved, but it will transition to the Available state when the publishing operation is complete. The web interface does not allow you to change the state if it is currently Publishing. • in the Available state, the artifact version can be retrieved. To confirm this state, the checkbox is "checked," and unchecking this box makes the artifact version unavailable. • in the Unavailable state, the artifact version cannot be retrieved. "Checking" the box changes the artifact version state to Available.

-

-

-

Repositories

This page displays all artifact repositories available to this ElectricFlow server.

Links and actions at the top of the table

- **Add Repository** - Use this link to go to the New Repository page to add another artifact repository.
- The "star" icon allows you to save this page to your Home page for quick access.

Column descriptions

Column Name	Description / Actions
Repository Name	<p>The name of the artifact repository.</p> <p>Select a repository name to go to the Edit Repository page if you need to modify that repository.</p> <p>Note: If you have multiple repositories and want to change the search order for retrieving artifacts, drag the icon (in the far left column) up or down to reposition an artifact to the order you need.</p>
Zone	The name of the zone where this repository resides.
URL	The server URL is in the form <code>protocol://host:port/</code> . Typically, the repository server is configured to listen on port 8200 for <code>https</code> requests, so a typical URL looks like <code>https://host:8200/</code>
Enabled	Select this checkbox to enable or disable this artifact repository. By disabling this repository, users will not be able to retrieve any artifacts stored in this repository.
Description	A plain text or HTML description previously supplied for this repository.
Actions	Delete - Use this link to delete the artifact repository on this row.

Repository - create new or edit existing repository

To create a new repository

Enter information in the fields as follows:

Field Name	Description
Name	Enter any name of your choice for this repository.
Description	<p>A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code>.</p>

Field Name	Description
URL	The server URL is in the form <code>protocol://host:port/</code> . Typically, the repository server is configured to listen on port 8200 for <code>https</code> requests, so a typical URL looks like <code>https://host:8200/</code> .
Zone	Use the drop-down menu to select a zone for this repository. If no zone is selected, this repository will reside in the <i>default</i> zone.
Enabled	Select this checkbox to enable this repository. If this repository is disabled ("unchecked"), users will not be able to retrieve any artifacts stored in this repository.

Click **OK** after filling-in the fields and to return to the Repositories page to see your new repository listed in the first column.

To edit a repository

You can:

- Use the **Access Control** link at the top of the table to set permissions on this repository
- Change the repository name
- Add a description or modify the existing description
- Modify the URL
- Select a different zone.
- Enable or disable the repository

Click **OK** after making any modifications and to return to the Repositories page.

Database Configuration

If you choose not to use the ElectricFlow built-in database, use this page to configure your MySQL, SQL Server, or Oracle database to communicate with ElectricFlow.

Note: The ElectricFlow built-in database is not recommended for production use.

Fill in the fields as follows:

Field Name	Description
Database Type	Select your database type from the drop-down menu. If "Built-in" is selected, no other information is required.

Field Name	Description
Database Name	Enter your database name.
Host Name	Enter the host name for your database server.
Port	Use the default port or enter a new port number.
Database Credentials	User Name—Accept the default "commander" user name or enter the appropriate user name for your database. Password—Enter the database password.

IMPORTANT: When you set the SQL server as the database, you can only use a non-named database.

Click **Save and Restart Server** after entering information in all fields.

Note: You may need to consult with your Database Administrator if you do not have all of the information required on this web page.

For more information, consult the *ElectricFlow Installation Guide*.

Defect Tracking Configurations

This page displays Defect Tracking configurations known to the ElectricFlow server—ElectricFlow integrates, using plugins, with numerous defect tracking systems.

Link at the top of the table

Use the **Create Configuration** link to create a new Defect Tracking configuration.

Column descriptions

Column Name	Description
Configuration Name	The name you provided to the configuration you created.
Description	This column describes the Defect Tracking configuration type.
Plugin	The name of the plugin where the Defect Tracking integration resides.
Actions	Edit - Click this link to modify an existing configuration. Delete - Click this link to remove the configuration.

Defect Tracking - create new or edit existing configuration

Use this page to define a new defect tracking configuration or modify an existing defect tracking configuration. A configuration is a collection of properties that define how ElectricFlow communicates with a particular defect tracking system.

After creating a defect tracking configuration, your entry will appear in the table on the Defect Tracking Configurations web page— to see this web page, select the Administration > Defect Tracking tabs.

To create a defect tracking configuration

From the Defect Tracking Type drop-down menu, select your defect tracking system—ElectricFlow integrates with numerous defect tracking systems. Each integration was created using plugin technology.

Click the **Submit** button and a set of fields is displayed to enter information to create your configuration.

Enter information in the fields as follows:

Field Name	Description
Configuration Name	Enter a unique name for this defect tracking configuration—any name you choose.
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code>
Login As	<p>User Name - This is the name ElectricFlow needs to use to communicate with your defect tracking system. For example, you may be using a special "read-only" user name similar to "Build" for your user name.</p> <p>Password - This is the password for the specified User Name.</p> <p>Retype Password - Type the password again.</p> <p>URL - This is the URL to use to connect to a JIRA server. For example, you might use something similar to: <code>http://10.10.10.10:8080</code> or <code>http://yourJIRAserver</code></p>

Click **Submit** to save your information.

Using the defect tracking integration

Go to Projects > select a project > select a procedure.

- To create a New Step for defect tracking, select the Plugin link.
- In the Choose Step panel, select Defect Tracking from the left pane, then select the defect tracking system you configured.
- The right-pane now shows the types of steps available for your configuration. Select the step you need and automatically go to the New Step page.
- On the New Step page, notice the Subprocedure section now contains the defect tracking integration you configured and the step you chose.
- Enter the remaining information to create your defect tracking step.

See the [Defect Tracking](#) Help topic for more information.

To edit an existing defect tracking configuration

Modify any of your defect tracking configuration information by "typing-over" any previously entered information.

Click **Submit** to save your modified defect tracking information.

Defect Tracking Reports

This is a report of URLs to JIRA defects associated with the ElectricFlow job.

The report was compiled by searching job properties for JIRA defect IDs then querying the configured JIRA server for the current defect state.

For more information about Defect Tracking, see the main [Defect Tracking](#) Help topic.

If you would like to create a defect tracking configuration, go to Administration > Defect Tracking and click the **Create Configuration** link to see the New Defect Tracking Configuration web page. Access the Help link on that page for more information.

Email Notifier - create new or edit existing email notifier

Use this page to set up email notifiers.

Note: You must have already set up an Email Configuration so ElectricFlow knows which email server to use when sending any notifications. If you have not configured an email server to communicate with ElectricFlow, click the Administration > Email Configuration tabs to do this configuration now.

For jobs and job steps - two types of email notifiers:

- On Start Notifier - sends an email when a job or job step starts.
- On Completion Notifier - sends an email when a job or job step completes.

For workflow states - three types of email notifiers:

- On Enter Notifier - sends an email when the state becomes the workflow's active state.
- On Start Notifier - sends an email after the state's subjob or subworkflow starts. If no subjob or subworkflow is defined for that state, these notifiers will not be sent.
- On Completion Notifier - sends an email after the state's subjob or subworkflow completes. If no subjob or subworkflow is defined for that state, these notifiers will not be sent.

Using email notifiers

- Attaching an Email Notifier to a procedure results in a corresponding Email Notifier associated with the job that is created when that procedure is executed.
- Attaching an Email Notifier to a procedure step results in a corresponding Email Notifier associated with the job step created when the procedure referencing the procedure step is executed.
- Attaching an Email Notifier to a state definition results in a corresponding Email Notifier associated with the state that is created when that workflow is executed.

To create a new email notifier

Enter information in the fields as follows:

Field Name	Description
Name	This name can be an arbitrary text string.
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code>
Condition	Use the pull-down menu to select the type of condition you need for this email notifier. Edit the auto-supplied condition in the text box or add a completely new script for your purpose. The condition specifies whether the notifier should send a message depending on the result of a property expansion. If the result is empty or non-zero, the message is sent. If the result is "0", the message is not sent

Field Name	Description
Formatting Template	<p>Use the pull-down menu to select from a list of global, ready-to-use formatting templates. Depending on the type of email notifier you are creating, the available template choices in the drop-down menu will be different. Make sure the content is formatted correctly, i.e., no illegal characters or spacing.</p> <p>To customize your template, edit the auto-supplied text in the Formatting Template text box or you can add a completely new script for your purpose. Any edits made in this text box will not be saved to the global template.</p> <p>Note: See the "Email notifier templates" section at the end of this Help topic for a list of available templates and instructions for making global modifications to these templates.</p> <p>To create a custom template, the basic structure is:</p> <ul style="list-style-type: none"> • zero or more email header lines • blank line • message body <p>The template undergoes property expansion in a <code>job</code>, <code>jobStep</code>, or <code>state</code> context. In addition to normal property paths, an additional event object is accessible via <code>/myEvent/</code>.</p> <p>The following example is a sample formatting template for a job:</p> <pre>Subject: Job '\${jobName}' from procedure '\${procedureName}' \${/myEvent/type} - ElectricFlow notification cc: owner@example.org bcc: admin@example.org ElectricFlow Notification - from email notifier '\${/myEvent/notifier}' Job '\${jobName}' \${/myEvent/type} at: \${/myEvent/time} Project: \${projectName} Procedure: \${procedureName} Started: \${start} Completed: \${finish} Directory Name: \${directoryName} Launched by User: \${launchedByUser} Outcome: \${outcome} Error Code: \${errorCode} Error Message: \${errorMessage}</pre> <p>This example demonstrates how you can send this notification using CC or</p>

Field Name	Description
	BCC fields in addition to specified Destinations. Because headers are interpreted by SMTP, normal email rules apply.
Email Configuration	Click inside this field or start typing to bring up a list of possible email configuration names. An email notifier that does not specify an email configuration will use the configuration named 'default' if it exists.
Destinations	A space-separated list of valid email addresses, email aliases, or ElectricFlow user or group names, or a property reference that expands into such a list, or you can enter an LDAP DL name (group name).

Click **OK** to save your email notifier configuration.

To edit an existing email notifier

Modify or add information to any of the fields and click **OK** to save your changes.

Email notifier templates

The table below lists global email notifier templates for your use. Each template name is a link to an example of the template output and the script.

Global templates are stored as property sheets under the `"/server/ec_notifierTemplates"` property sheet. You can modify template properties or add new global templates if you have modify privileges on the server property sheet. An email notifier will be updated if changes are made to its global template.

Template Name	Format	Usage	Description
Job Summary text notification	Plain text	Procedure	Formatting template for Procedure starting/completion notifications using the long form of property names (for example, <code>\$/myJob/jobName</code>)
Step summary text notification	Plain text	Procedure step	Formatting template for Procedure Step starting/completion notifications using the long form of property names (for example, <code>\$/myJobStep/jobStepId</code>)
State summary text notification	Plain text	State	Formatting template for State notifications using the long form of property names (for example, <code>\$/myWorkflow/workflowName</code>)
Job summary HTML notification	HTML	Procedure	HTML formatting template for Procedure starting/completion notifications
Step summary HTML notification	HTML	Procedure step	HTML formatting template for Procedure Step starting/completion notifications displaying information about the Job and the Job Step

Template Name	Format	Usage	Description
All steps HTML notification	HTML	Procedure step	HTML formatting template for Procedure Step starting/completion notifications displaying information about the Job and many Job Steps
Approval request HTML notification	HTML	State	HTML formatting template for State notifications displaying information about the workflow and all of its states
Workflow summary HTML notification	HTML	State	HTML formatting template for State notifications providing the recipient with links to view or take a manual transition

Email Configurations

This page displays all previously configured email configurations.

- Click on a Configuration Name to go to the Edit Email Configuration web page to modify an existing email configuration.
- Click on the Add Configuration link to create new email configuration.
- Action column - This column contains two links to Copy or Remove an Email Configuration.

Notice this page also lists the email server name and who the email notification was from.

Email Configuration - create new or edit existing email configuration

Use this web page to specify an email server to ElectricFlow. If you do not create an Email Configuration, you will not be able to send Email Notifications to individuals or groups.

If you have multiple users or groups in remote locations that use a different mail server, create additional Email Configurations to accommodate those locations if they need to receive ElectricFlow notifications.

To create a new email configuration

Enter information in the fields as follows:

Field Name	Description
Name	Enter a unique name to identify the Email Configuration.
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code>
Mail Protocol	Either SSMTP or SMTP
Mail Host	The name of the mail server.
Mail Port	The port number for the mail server, but may not need to be specified. Protocol software determines the default value (25 for SMTP and 465 for SSMTP). Specify a value for this argument when a non -default port is used.
Mail From	An individual name or a generic name like "ElectricFlow" or "build."
Mail Username	An individual name or a generic name like "ElectricFlow."
Mail Password	The password for the mail user name account.

Click **Save** after filling-in all fields.

Click the **Test** button to run a test to make sure your email configuration works. Enter an email address in the pop-up box and click **Send**.

To edit an existing email configuration

Use the **Access Control** link at the top of the page to add or change existing permissions.

You can modify any of the information already supplied or add new information.

Click **OK** after completing your edits.

Email Notifier - create new or edit existing email notifier

Use this page to set up email notifiers.

Note: You must have already set up an Email Configuration so ElectricFlow knows which email server to use when sending any notifications. If you have not configured an email server to communicate with ElectricFlow, click the Administration > Email Configuration tabs to do this configuration now.

For jobs and job steps - two types of email notifiers:

- On Start Notifier - sends an email when a job or job step starts.
- On Completion Notifier - sends an email when a job or job step completes.

For workflow states - three types of email notifiers:

- On Enter Notifier - sends an email when the state becomes the workflow's active state.
- On Start Notifier - sends an email after the state's subjob or subworkflow starts. If no subjob or subworkflow is defined for that state, these notifiers will not be sent.
- On Completion Notifier - sends an email after the state's subjob or subworkflow completes. If no subjob or subworkflow is defined for that state, these notifiers will not be sent.

Using email notifiers

- Attaching an Email Notifier to a procedure results in a corresponding Email Notifier associated with the job that is created when that procedure is executed.
- Attaching an Email Notifier to a procedure step results in a corresponding Email Notifier associated with the job step created when the procedure referencing the procedure step is executed.
- Attaching an Email Notifier to a state definition results in a corresponding Email Notifier associated with the state that is created when that workflow is executed.

To create a new email notifier

Enter information in the fields as follows:

Field Name	Description
Name	This name can be an arbitrary text string.
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code>
Condition	Use the pull-down menu to select the type of condition you need for this email notifier. Edit the auto-supplied condition in the text box or add a completely new script for your purpose. The condition specifies whether the notifier should send a message depending on the result of a property expansion. If the result is empty or non-zero, the message is sent. If the result is "0", the message is not sent

Field Name	Description
Formatting Template	<p>Use the pull-down menu to select from a list of global, ready-to-use formatting templates. Depending on the type of email notifier you are creating, the available template choices in the drop-down menu will be different. Make sure the content is formatted correctly, i.e., no illegal characters or spacing.</p> <p>To customize your template, edit the auto-supplied text in the Formatting Template text box or you can add a completely new script for your purpose. Any edits made in this text box will not be saved to the global template.</p> <p>Note: See the "Email notifier templates" section at the end of this Help topic for a list of available templates and instructions for making global modifications to these templates.</p> <p>To create a custom template, the basic structure is:</p> <ul style="list-style-type: none"> • zero or more email header lines • blank line • message body <p>The template undergoes property expansion in a <code>job</code>, <code>jobStep</code>, or <code>state</code> context. In addition to normal property paths, an additional event object is accessible via <code>/myEvent/</code>.</p> <p>The following example is a sample formatting template for a job:</p> <pre>Subject: Job '\${jobName}' from procedure '\${procedureName}' \${/myEvent/type} - ElectricFlow notification cc: owner@example.org bcc: admin@example.org ElectricFlow Notification - from email notifier '\${/myEvent/notifier}' Job '\${jobName}' \${/myEvent/type} at: \${/myEvent/time} Project: \${projectName} Procedure: \${procedureName} Started: \${start} Completed: \${finish} Directory Name: \${directoryName} Launched by User: \${launchedByUser} Outcome: \${outcome} Error Code: \${errorCode} Error Message: \${errorMessage}</pre> <p>This example demonstrates how you can send this notification using CC or</p>

Field Name	Description
	BCC fields in addition to specified Destinations. Because headers are interpreted by SMTP, normal email rules apply.
Email Configuration	Click inside this field or start typing to bring up a list of possible email configuration names. An email notifier that does not specify an email configuration will use the configuration named 'default' if it exists.
Destinations	A space-separated list of valid email addresses, email aliases, or ElectricFlow user or group names, or a property reference that expands into such a list, or you can enter an LDAP DL name (group name).

Click **OK** to save your email notifier configuration.

To edit an existing email notifier

Modify or add information to any of the fields and click **OK** to save your changes.

Email notifier templates

The table below lists global email notifier templates for your use. Each template name is a link to an example of the template output and the script.

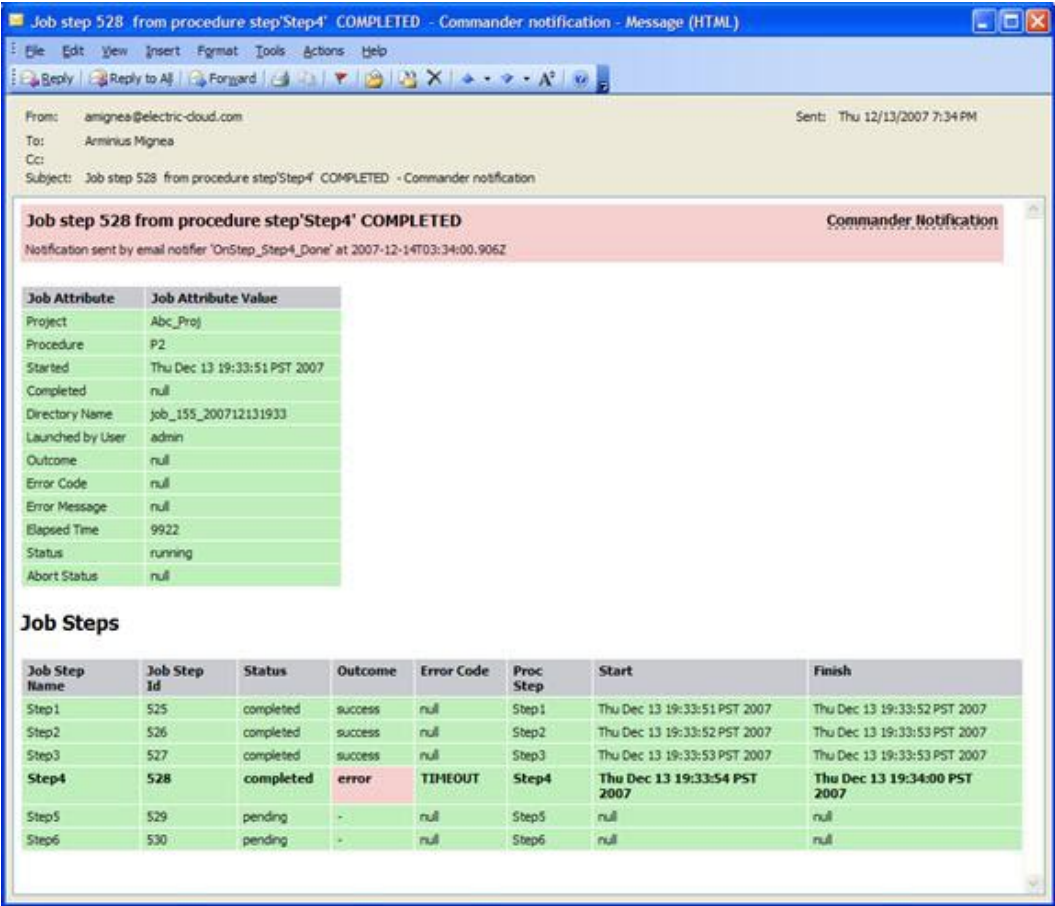
Global templates are stored as property sheets under the `"/server/ec_notifierTemplates"` property sheet. You can modify template properties or add new global templates if you have modify privileges on the server property sheet. An email notifier will be updated if changes are made to its global template.

Template Name	Format	Usage	Description
Job Summary text notification	Plain text	Procedure	Formatting template for Procedure starting/completion notifications using the long form of property names (for example, <code>\$/myJob/jobName</code>)
Step summary text notification	Plain text	Procedure step	Formatting template for Procedure Step starting/completion notifications using the long form of property names (for example, <code>\$/myJobStep/jobStepId</code>)
State summary text notification	Plain text	State	Formatting template for State notifications using the long form of property names (for example, <code>\$/myWorkflow/workflowName</code>)
Job summary HTML notification	HTML	Procedure	HTML formatting template for Procedure starting/completion notifications
Step summary HTML notification	HTML	Procedure step	HTML formatting template for Procedure Step starting/completion notifications displaying information about the Job and the Job Step

Template Name	Format	Usage	Description
All steps HTML notification	HTML	Procedure step	HTML formatting template for Procedure Step starting/completion notifications displaying information about the Job and many Job Steps
Approval request HTML notification	HTML	State	HTML formatting template for State notifications displaying information about the workflow and all of its states
Workflow summary HTML notification	HTML	State	HTML formatting template for State notifications providing the recipient with links to view or take a manual transition

Email Notifier Template - Html_JobStepTempl_AllSteps.txt

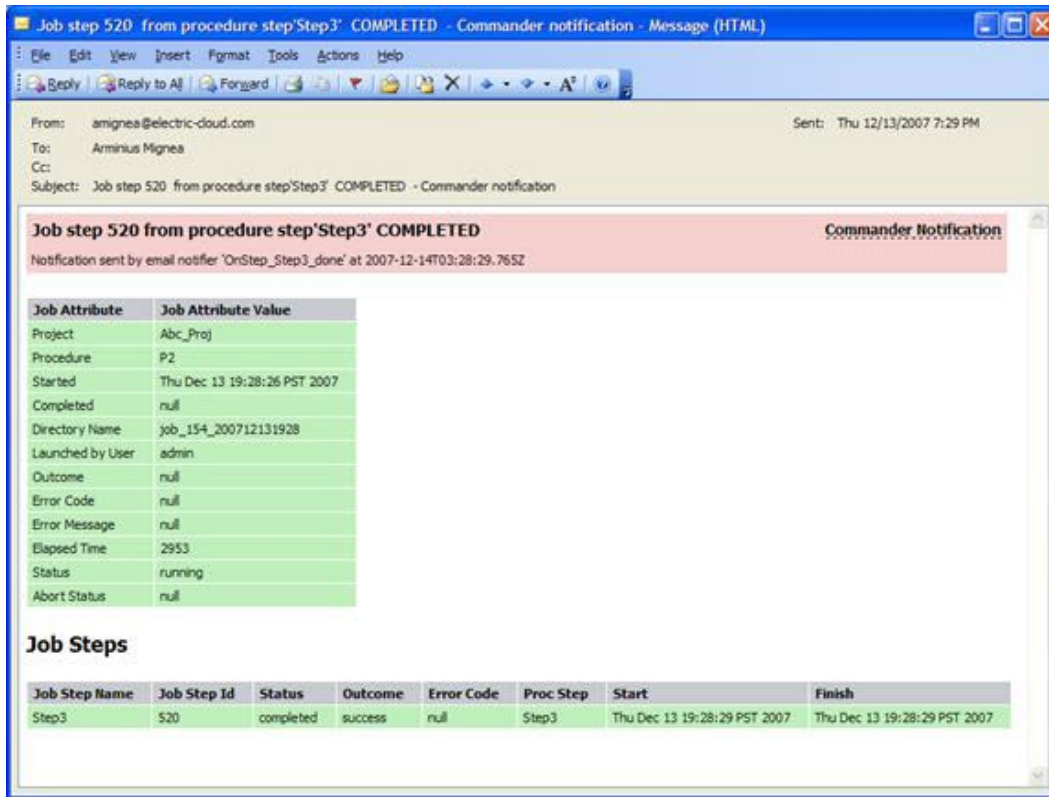
If you use this template, the following screen example is similar to how your email notifier will look.



1. Go to the corresponding template available in `<installDir>\src\samples\notifier`
2. Copy the script into a text editor to edit it for your purpose
3. "Copy and paste" your version into the Formatting Template field.

Email Notifier Template - Html_JobStepTempl_SingleStep.txt

If you use this template, the following screen example is similar to how your email notifier will look.

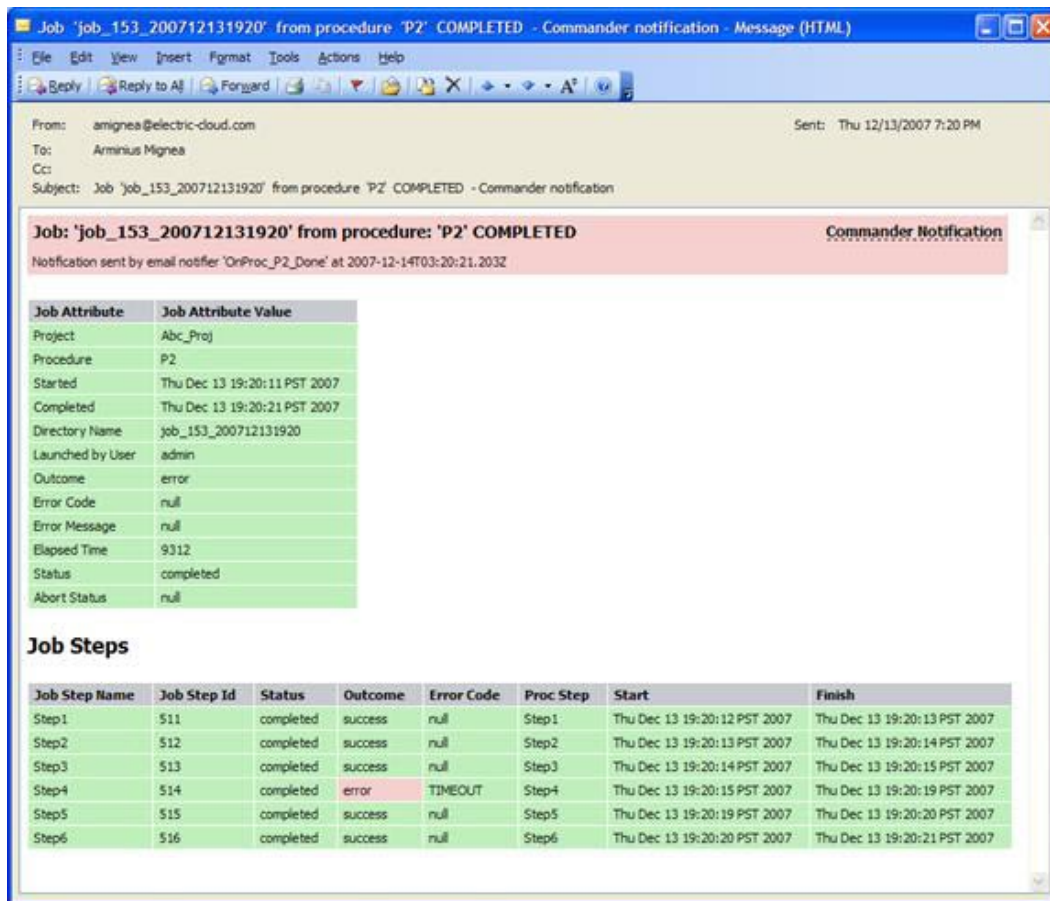


Note: In the following script, myJob.steps and myJob.jobSteps return an array of step names.

1. Go to the corresponding template available in `<installDir>\src\samples\notifier`
2. Copy the script into a text editor to edit it for your purpose
3. "Copy and paste" your version into the Formatting Template field.

Email Notifier Template - Html_JobTempl.txt

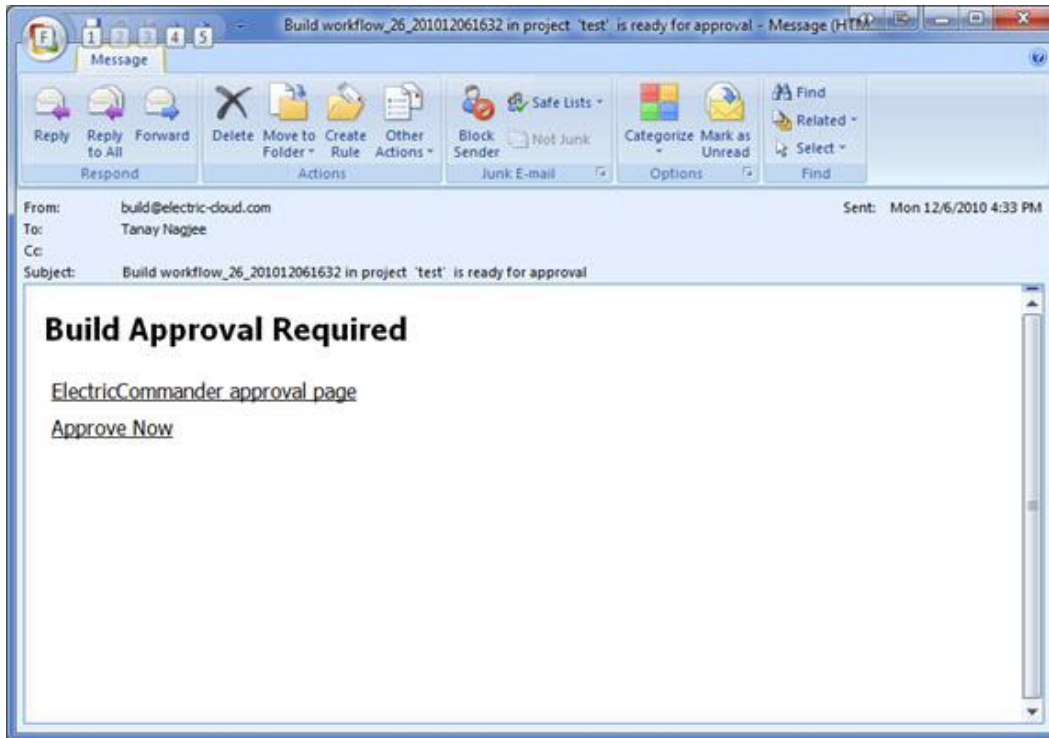
If you use this template, the following screen example is similar to how your email notifier will look.



1. Go to the corresponding template available in `<installDir>\src\samples\notifier`
2. Copy the script into a text editor to edit it for your purpose
3. "Copy and paste" your version into the Formatting Template field.

Email Notifier Template - Html_StateTemplate_ApproveWorkflow.txt

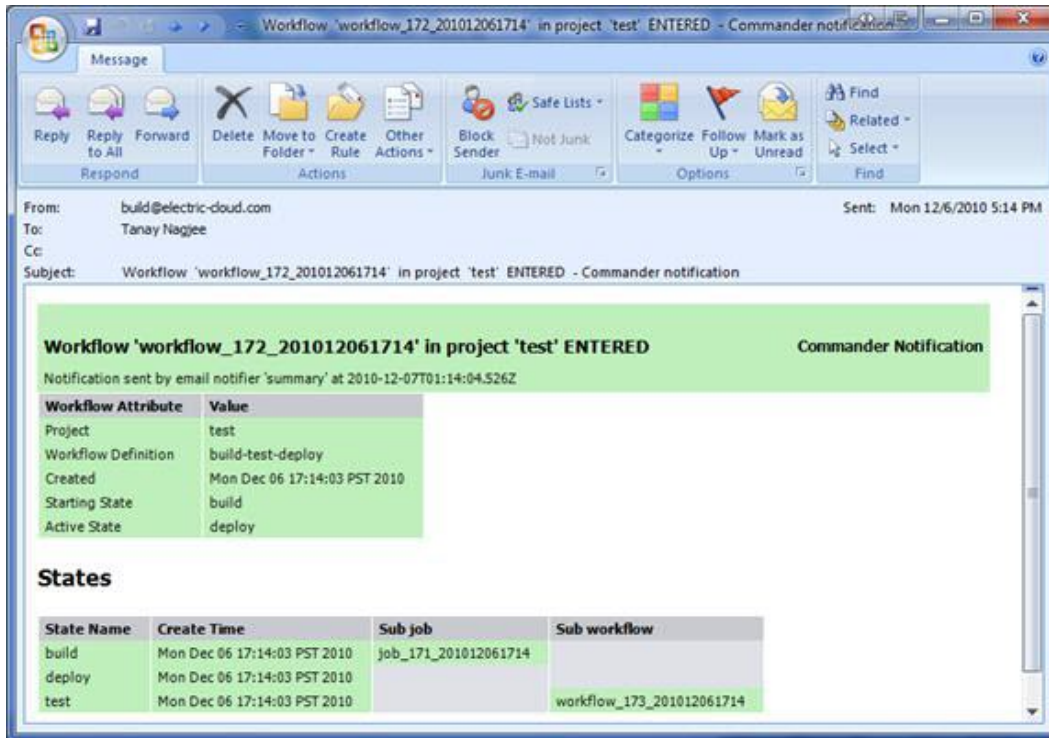
If you use this template, the following screen example is similar to how your email notifier will look.



1. Go to the corresponding template available in `<installDir>\src\samples\notifier`
2. Copy the script into a text editor to edit it for your purpose
3. "Copy and paste" your version into the Formatting Template field.

Email Notifier Template - Html_StateTemplate_FullWorkflow.txt

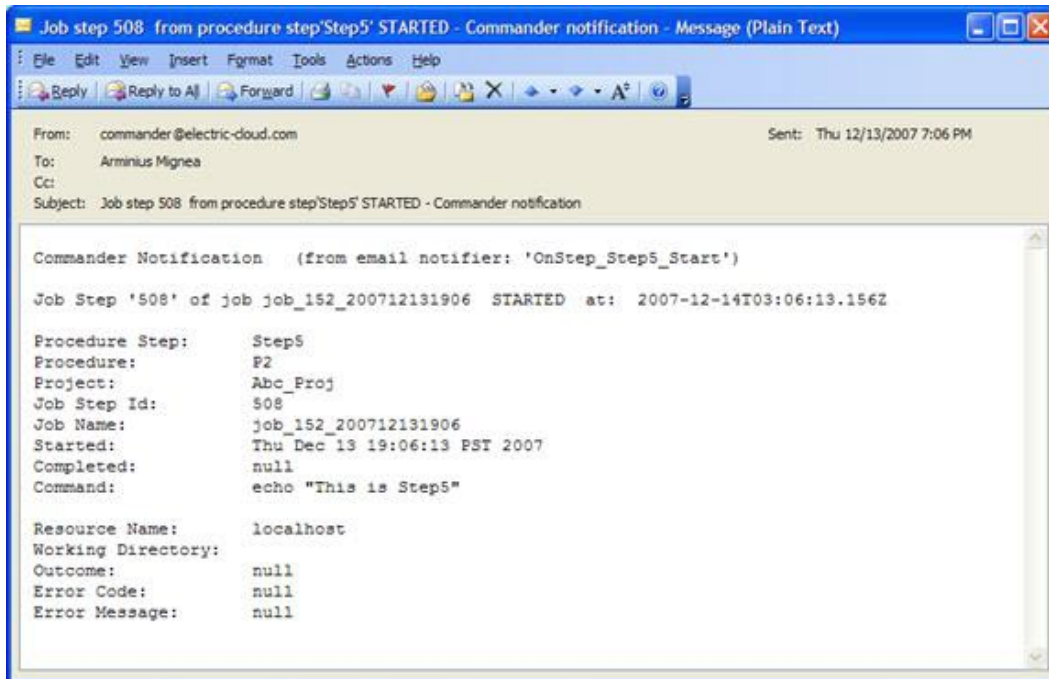
If you use this template, the following screen example is similar to how your email notifier will look.



1. Go to the corresponding template available in `<installDir>\src\samples\notifier`
2. Copy the script into a text editor to edit it for your purpose
3. "Copy and paste" your version into the Formatting Template field.

Email Notifier Template - JobStepTempl_FullProps.txt

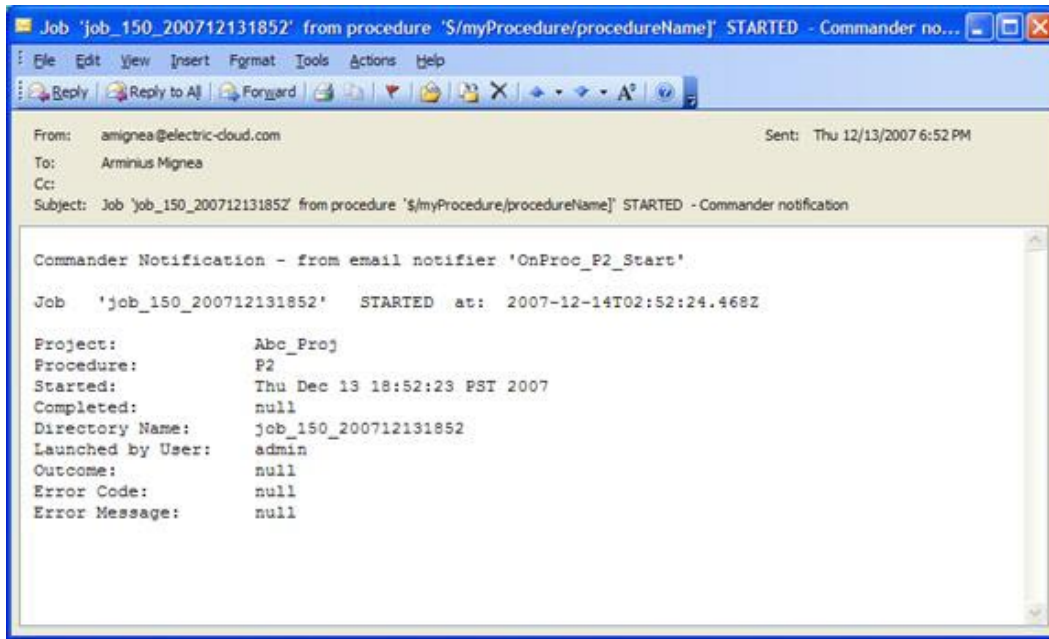
If you use this template, the following screen example is similar to how your email notifier will look.



1. Go to the corresponding template available in `<installDir>\src\samples\notifier`
2. Copy the script into a text editor to edit it for your purpose
3. "Copy and paste" your version into the Formatting Template field.

Email Notifier Template - JobTempl_FullProps.txt

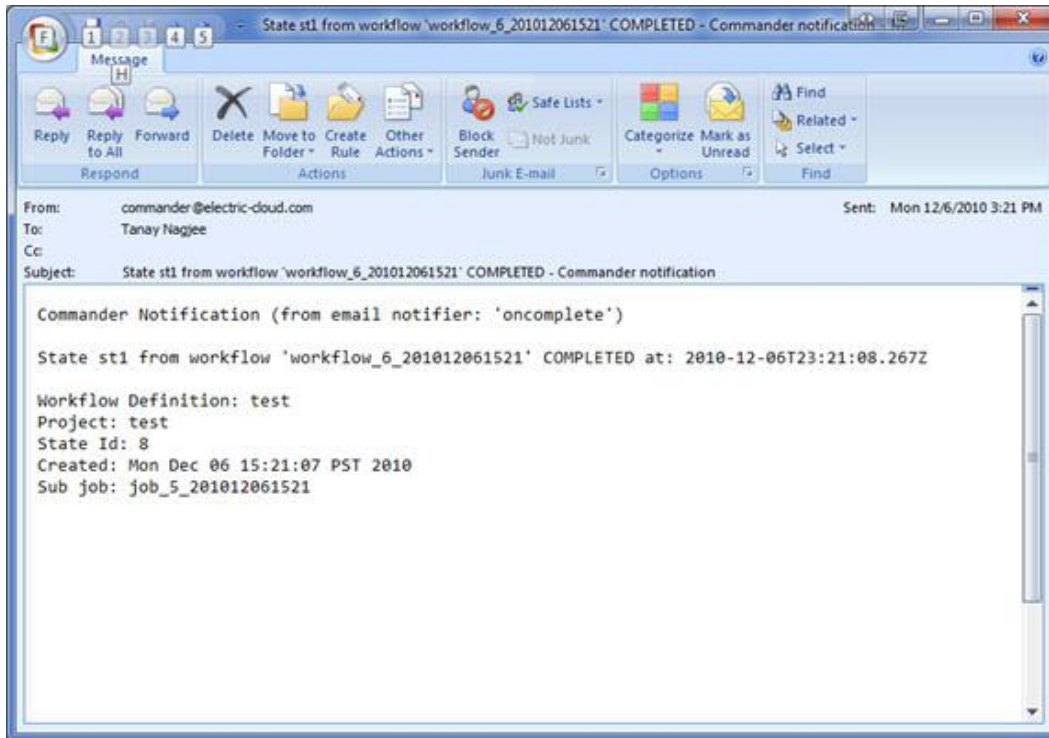
If you use this template, the following screen example is similar to how your email notifier will look.



1. Go to the corresponding template available in `<installDir>\src\samples\notifier`
2. Copy the script into a text editor to edit it for your purpose
3. "Copy and paste" your version into the Formatting Template field.

Email Notifier Template - StateTemplate_FullPropertyPaths.txt

If you use this template, the following screen example is similar to how your email notifier will look.



1. Go to the corresponding template available in `<installDir>\src\samples\notifier`
2. Copy the script into a text editor to edit it for your purpose
3. "Copy and paste" your version into the Formatting Template field.

Event Log

This page displays a log of events generated anywhere in the system, including jobs and workflows.

- To see only events for a single workflow, select the Workflows tab, then a workflow Name to go to the Workflow Details page and click the **View Log** link at the top of the page.
- To see only events for a single job, select the Jobs tab, then the Job name to go to the Job Details page and click the **View Log** link at the top of the page.

Note: When running a job in a cluster, the name of the server managing the job is recorded in the job-log. If the management of the job moves to a different node (if the current node crashes for example) the change is logged into the job log as well.

- To see only events for a specific object, select the Search tab to go to the Define Search page.

For example:

You might want to select the Object Type, "Log Entry", then click the **Add Intrinsic Filter** link. Select the down-arrow where you see "Container" auto-populated and select "Container Type". Use the "equals" operator, then select the next down-arrow to choose an object. Click **OK** to start the search.

Note:

From the Administration tab, the default view for this Event Log page is the warning (WARN) level.

For workflow and job event logs, the default view from their respective pages is the information (INFO) level.

Column descriptions

Time - This is the time the event was logged.

Severity - Severity can be either INFO, WARN, or ERROR. Use the pull-down menu at the top of the table to change between these views.

User - The user associated with the event—the user of the session at the time the event was generated. Clicking on a project link in this column takes you to the Project Details page for that project.

Container - Typically, this is the type and name of the workflow or job with a corresponding ID. Clicking on a link in this column takes you to the Job Details page for that job or workflow.

Subject - The object associated with the message. Clicking on a link in this column takes to you the source for this object.

Message - The message text will either be informational or display the warning or error message. The message may contain important information about a resource or workspace issue.

Configuring the Event Log

Because log entries can accumulate in large numbers, you can configure the Event Log for auto-deletion. By default, Event Log retention is 30 days. The ElectricFlow server automatically marks old log entries for deletion and the background deleter cleans out old log entries along with its other tasks.

To change the default settings, go to Administration > Server > Settings.

Modify the following two fields:

Event log retain time - Default is 30 days (in "minute" units).

Note: Setting the "retain" period to "0" disables the automatic deletion mechanism, allowing you to use an external cleanup script to implement a custom cleanup policy.

Maximum background delete delay - Default is 3600 seconds (or one hour).

Click **OK** to save your changes.

Automation Platform Home Page

[Job Configurations](#)

[Shortcuts](#)

[Jobs Quick View](#)

[Reports](#)

Overview

This page provides a convenient console for running jobs and viewing results.

- This web page is your ElectricFlow Dashboard for tracking project health.
- You can customize the page to display only the jobs that interest you and to provide one-click access to the jobs you run frequently. Each of the Home page sections can be customized.
- The Reports section allows you to see thumbnail report images that are updated each time the report is generated.
- This page provides shortcuts for quick access to pages you use most frequently in the automation platform or anywhere on the web.

IMPORTANT: If you are using or plan to use the EC-Homepage plugin to share your Home page sections, you will not see some links normally available to you. For example, if a particular Jobs Quick View category is shared, you **cannot** Delete or Modify that category. You can perform these functions only if the category belongs to you alone.

For information on sharing your Home page configuration, see the "[Home page configurations](#)" section in the "[Customizing the ElectricFlow Platform UI](#) on page 583" Help topic.

Job Configurations

Procedures in ElectricFlow can contain complex sets of parameters—making it difficult to remember the correct parameters for a particular situation and tedious to re-enter those parameters every time the procedure is invoked. Job Configurations provide a one-click solution to this problem. When you create a job configuration, you enter all the information needed to run a procedure, including parameters and/or a credential. All job configurations are displayed here on the Home page. Invoke a particular configuration by clicking its name in the Job Configurations section.

Create Job Configurations three ways

- On the Home page, create a job configuration from "scratch" by clicking the **Create** link in the Job Configurations section.
 - In the Create Configuration pop-up menu, select the project and procedure you want to use for creating this configuration.
- From the Job Details page for a previously invoked job, click the **Save Configuration** link at the top of the page.

Your saved job configuration is displayed on your Home page.
- From the Edit Schedule page, click the **Save Configuration** link at the top of the page.

Your saved configuration is displayed on your Home page.

Shortcuts

Use shortcuts to save frequently visited ElectricFlow web pages, so those pages are immediately accessible. You can create a shortcut to any page on the web also.

Create Shortcuts two ways

- Mouse-over the "star" icon at the top of any automation platform page and click "Add current page" to add that page to the shortcut list. Mouse-over the star icon again and click "Remove current page" to remove the shortcut for that page. The star icon is yellow for pages saved as a shortcut and gray for those pages not saved as a shortcut.
- Click the **Create** link in the Shortcut section and provide a name and URL to create a shortcut.

To modify or update a shortcut, click the **Edit** link adjacent to the shortcut you want to change.

Shortcuts can be accessed conveniently from any automation platform web page. Mousing-over the star icon displays a list of shortcuts saved by the current user. Click on a shortcut name to view the page.

Note: The Shortcut section may contain static entries that cannot be deleted. And if you "share" your Home page, the Edit link will not be available for shared items.

Jobs Quick View

Perhaps only a few jobs on this server are of interest to you. For example, you may care about jobs you launch manually and official builds for the products you work on, but you may not care about production builds for other products or personal jobs for other users.

The Jobs Quick View allows you to define job categories that are interesting to you.

The Home page displays the most recent jobs in each category, and you can easily click-through to get more details about any of those jobs. For example, clicking on a job name takes you to that job's Job Details page.

Create a job category

- Click the **Add Category** link in the Jobs Quick View section.
After creating a category, results are displayed on the Home page. Clicking the **Details** link displays a summary to the right of the category. In addition to job status and other diagnostic information, the summary displays running steps and failed steps (containing errors and warnings).
- Click the **Modify** link to edit the Jobs Quick View category or the **Delete** link to remove that job from the Jobs Quick View category.
- Click on the **Details** link to see job summary information—the summary remains visible, regardless of mouse location, until you click somewhere else on the page.

Note: If you "share" your Home page, the Modify and Delete links will **not** be available for shared items.

Reports

You can configure reports you would like to see on a regular basis and display a "thumbnail" report graphic in this section.

- Click the **Add Report** link to go to the New Reports page.
After filling-in the information on the New Reports page, you will see a thumbnail view of your report (after it runs) on your Home page.
Note: If the drop-down menu on this page is empty, click the **Help** link on the New Report page for more information.
- Click the Collapse/Expand (+/-) box to see the full thumbnail report or just the report title.

- Click the **Rename** link if you need to rename your report.
- Click the **Remove** link if you need to remove this report from your Home page.

Note: If you "share" your Home page, the Rename and Remove links will not be available for shared items.

Automation Platform Home Page

[Job Configurations](#)

[Shortcuts](#)

[Jobs Quick View](#)

[Reports](#)

Overview

This page provides a convenient console for running jobs and viewing results.

- This web page is your ElectricFlow Dashboard for tracking project health.
- You can customize the page to display only the jobs that interest you and to provide one-click access to the jobs you run frequently. Each of the Home page sections can be customized.
- The Reports section allows you to see thumbnail report images that are updated each time the report is generated.
- This page provides shortcuts for quick access to pages you use most frequently in the automation platform or anywhere on the web.

IMPORTANT: If you are using or plan to use the EC-Homepage plugin to share your Home page sections, you will not see some links normally available to you. For example, if a particular Jobs Quick View category is shared, you **cannot** Delete or Modify that category. You can perform these functions only if the category belongs to you alone.

For information on sharing your Home page configuration, see the "[Home page configurations](#)" section in the "[Customizing the ElectricFlow Platform UI](#) on page 583" Help topic.

Job Configurations

Procedures in ElectricFlow can contain complex sets of parameters— making it difficult to remember the correct parameters for a particular situation and tedious to re-enter those parameters every time the procedure is invoked. Job Configurations provide a one-click solution to this problem. When you create a job configuration, you enter all the information needed to run a procedure, including parameters and/or a credential. All job configurations are displayed here on the Home page. Invoke a particular configuration by clicking its name in the Job Configurations section.

Create Job Configurations three ways

- On the Home page, create a job configuration from "scratch" by clicking the **Create** link in the Job Configurations section.
 - In the Create Configuration pop-up menu, select the project and procedure you want to use for creating this configuration.
- From the Job Details page for a previously invoked job, click the **Save Configuration** link at the top of the page.
Your saved job configuration is displayed on your Home page.
- From the Edit Schedule page, click the **Save Configuration** link at the top of the page.
Your saved configuration is displayed on your Home page.

Shortcuts

Use shortcuts to save frequently visited ElectricFlow web pages, so those pages are immediately accessible. You can create a shortcut to any page on the web also.

Create Shortcuts two ways

- Mouse-over the "star" icon at the top of any automation platform page and click "Add current page" to add that page to the shortcut list. Mouse-over the star icon again and click "Remove current page" to remove the shortcut for that page. The star icon is yellow for pages saved as a shortcut and gray for those pages not saved as a shortcut.
- Click the **Create** link in the Shortcut section and provide a name and URL to create a shortcut.

To modify or update a shortcut, click the **Edit** link adjacent to the shortcut you want to change.

Shortcuts can be accessed conveniently from any automation platform web page. Mousing-over the star icon displays a list of shortcuts saved by the current user. Click on a shortcut name to view the page.

Note: The Shortcut section may contain static entries that cannot be deleted. And if you "share" your Home page, the Edit link will not be available for shared items.

Jobs Quick View

Perhaps only a few jobs on this server are of interest to you. For example, you may care about jobs you launch manually and official builds for the products you work on, but you may not care about production builds for other products or personal jobs for other users.

The Jobs Quick View allows you to define job categories that are interesting to you.

The Home page displays the most recent jobs in each category, and you can easily click-through to get more details about any of those jobs. For example, clicking on a job name takes you to that job's Job Details page.

Create a job category

- Click the **Add Category** link in the Jobs Quick View section.
After creating a category, results are displayed on the Home page. Clicking the **Details** link displays a summary to the right of the category. In addition to job status and other diagnostic information, the summary displays running steps and failed steps (containing errors and warnings).

- Click the **Modify** link to edit the Jobs Quick View category or the **Delete** link to remove that job from the Jobs Quick View category.
- Click on the **Details** link to see job summary information—the summary remains visible, regardless of mouse location, until you click somewhere else on the page.

Note: If you "share" your Home page, the Modify and Delete links will **not** be available for shared items.

Reports

You can configure reports you would like to see on a regular basis and display a "thumbnail" report graphic in this section.

- Click the **Add Report** link to go to the New Reports page.
After filling-in the information on the New Reports page, you will see a thumbnail view of your report (after it runs) on your Home page.
Note: If the drop-down menu on this page is empty, click the **Help** link on the New Report page for more information.
- Click the Collapse/Expand (+/-) box to see the full thumbnail report or just the report title.
- Click the **Rename** link if you need to rename your report.
- Click the **Remove** link if you need to remove this report from your Home page.

Note: If you "share" your Home page, the Rename and Remove links will not be available for shared items.

Job Configuration

To create a new job configuration

Enter information in the fields as follows:

Name - Type a name you choose for the job configuration.

Procedure - Displays the current project and procedure. Click **Change** to use the pop-up menu to select a new project/procedure.

Parameters

Depending on the Project and Procedure you selected, a set of Parameters is displayed.

Enter all parameter values or just the required values if differentiated.

Advanced

Priority:

Use the drop-down menu to select the priority for this run procedure. Available priorities are: low, normal (default), high, or highest.

- You can select the job's priority here, on Run Procedure web page, or when you schedule the procedure to run at a regular time or interval. When a job is launched, its priority cannot be changed.

- Priorities take effect when two or more job steps in different jobs are waiting for the same resource. When the resource is available, it will be used by the job step that belongs to the job with the highest priority. If the priority level is the same, the resource will be used by the job step that belongs to the job with the lowest job ID number. If the job steps are in the same job, the resource will be used first by the step with the lowest job step ID number.
- If using ectool, the priority can be set by passing "--priority=<low|normal|high|highest>" to a `runProcedure`, `createSchedule`, or `modifySchedule` operation.

Note: To raise the job priority level above "normal," the user who launches the procedure must have execute permission on the system-level access control list for priorities.

Impersonation:

Click one of the three "radio" buttons to determine the default login user to use when the configuration is run.

- Use pre-defined credential - If you select this option, ElectricFlow looks for a credential first in the top-level procedure and then in its project. If it finds a credential, it uses the login from that credential as the default for the job. If there is no credential in either place, by default, the job's steps run as the user under which the agent for the step is running.
- Use specific credential - If you select this option, you can choose a particular credential from those defined for the project containing the procedure - the login user from that credential is used as the default for the job.
- Use a specific user - If you select this option, you can enter a login name. When the configuration is run, you will be asked to enter the password for that account. The specified login will be used as the default for the job.

For more information on credentials and user impersonation, see the [Credentials and User Impersonation](#) Help topic.

Click **OK** when all selections are complete. Return to the Home page to see your new job configuration displayed.

To edit an existing job configuration

Change or modify any of your previously specified information.

Click **OK** when all selections are complete.

Return to the Home page to see your edited job configuration displayed.

Shortcuts

Use this page to create a **New Shortcut** or **Edit** an existing Shortcut.

Shortcuts appear on your Home page and provide quick access to other web pages, either in ElectricFlow or elsewhere.

- To create a new shortcut, type-in a shortcut name and the URL.
- To edit an existing shortcut, re-type the new shortcut name or URL (or both).

The easiest way to create a shortcut to an ElectricFlow automation platform page, such as the page for a particular project or procedure, is to navigate to that page and then click the "star" icon in the upper-right corner. This action automatically creates a shortcut and the star turns yellow to indicate a shortcut is in effect for the page.

If you click on the star icon again, the shortcut for that page is canceled.

Click **OK** to save your new or edited information and see your new/edited shortcut on the Home page.

Jobs Quick View

A quick view category provides a convenient way to select a group of jobs that interest you, such as "all production builds for product xyz version 3.6" or "all jobs I invoke manually." ElectricFlow displays the most recent jobs for each category on your Home page so you can review them easily.

To create a new jobs quick view category

Enter information in the fields as follows:

Name—Type a name for your new job category.

Number of Jobs—Type the maximum number of jobs you want to see in your quick view summary. The summary displays the most recent applicable jobs.

Include Last Success—Select this check box if you want to see the last successful job in this category, even if it was not one of the most recent jobs.

Two Filter categories:

- **Intrinsic filters**—These filters provide a convenient way to access certain well-defined fields for jobs.
- **Custom filters**—These filters allow you to access a much broader range of values, including custom properties. Any values accessible through an intrinsic filter can be checked using a custom filter also (though not as conveniently).

Each filter specifies three values: the name of a particular property to check, a value it should be compared with, and relational operators such as "equals" or "less than." For example, if you select "Outcome" in an intrinsic filter with operator "not equals" and value "Success," the filter selects jobs that did not complete successfully because they had errors or warnings.

For an intrinsic filter, select a property by choosing one of several predefined properties from a list. For a custom filter, enter a property name. This name can be either the name of an intrinsic property for the job (such as "status" or "procedureName"), or the name of a property defined in the global property sheet for that job, or the name of a parameter for that job.

For the relational operator, select one of the values in the list. The behavior of the relational operators is determined by the underlying database. In most cases, comparisons are done using string comparison; operators such as "less than" may not be useful on numbers because "12" is considered less than "9". For

some built-in properties, the database treats values according to a specific type such as number or date, so comparisons are implemented in the way appropriate for that type.

The relational operator "like" invokes an SQL wildcard comparison where "%" is the wildcard character. For example %test% matches "first test" and "tests failed" but not "Test 44".

Note: Job Quick View filters allow property path references. However, property path references are not allowed in search parameters on the Search page.

Click **OK** after making your selections to save your new job category and see it on the Home page.

To edit an existing jobs quick view category

Keep or change any of the previously entered information.

See the Filter information above to add or change existing custom or intrinsic filters.

Click **OK** after making your selections to save your modified job category and see it on the Home page.

New Report

Enter information in the following fields to define a new report to display on your Home page.

Field Name	Description
Project	The name of the project that contains the report you want to view.
Report	The report name you entered in the Report Title field when you configured this report. If you did not specify a "thumbnail" view when you defined your report, you will not see it here.
Title	This is the report title you want to see on your Home page to identify this report.

Click **OK** to continue.

After the report runs, the report you specified will be displayed on your Home page and updated whenever the report is generated again.

To populate the drop-down menu

- Go to the Project Details page for the project containing the reports you want to view.
- Select the Report tab.
- Click the **Create Report** link.
- Check the **Create thumbnail?** check box, which enters this report's name in the New Reports drop-down menu.

Note: All reports, **except** the Procedure Usage report, are available as "thumbnail" reports on your Home page.

Jobs

This page displays all jobs in the ElectricFlow system, both running and completed jobs. You can:

- Sort the Job, Status, Elapsed Time, and Start Time columns by clicking on the column name.
- Define a search to see only the jobs you choose.

Links at the top of the table

- **New Search** - click this link to go to the Define Search page to select a specific set or range of jobs to view.
- **Edit Search** - (available after a "search" is defined) click this link to redefine your search.
- **Refresh Results** - click this link to see updated status for running jobs.

Column descriptions

Job - click on Job to sort the column alphabetically or click on a job name to go to that job's Job Details page.

Status - click on Status to sort this column by Running, Success, Warning, Error, or Aborted.

Priority - displays the job's priority set by a "run procedure" command or by the job's schedule.

Procedure - in this column you can click on a project name to go to the Project Details page or click on the procedure name to go to the Procedure Details page for that procedure.

Launched By - click on a name in this column to go to the page for the schedule (Edit Schedule page) or the user that ran the job.

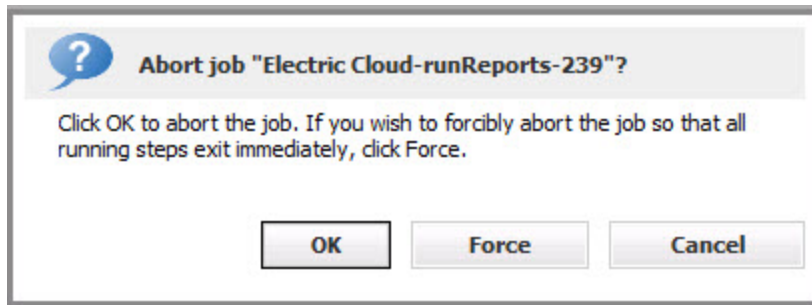
Elapsed Time - click Elapsed Time to sort the time values from longest to shortest time, or the reverse.

Start Time - click Start Time to sort this column from the start time of the first job to the start time of the most recent job, or the reverse.

Actions - use this column to Abort a running job or Delete a completed job.

- Aborting a job requires the execute privilege on the job—not just the modify privilege.

When you click Abort, the following dialog box appears:



Select either **OK** or **Force** to abort the job, or **Cancel** if you change your mind.

- Deleting a job on this page causes removal of job information from the ElectricFlow database, but information in the job's on-disk workspace area is not affected. You must delete workspace information manually.

Tips

If you would like to select a subset of the jobs, use the **Search** link at the top of the page.

From your Home page, you can add a specific "Search" for quick access on a regular basis.

Also notice: If you use RSS, an active RSS icon is provided in Windows Explorer on this page for your convenience. If you use Firefox, click **Bookmarks** > Subscribe to this page to display the feed. You can then add the feed URL to a viewer of your choice.

Job Step Details

This page displays detailed information about a job step.

Links and actions at the top of the table

- **Access Control** - Use this link to set privileges for this job step. For more information, see the Access Control Help topic.
- "star" icon - click this icon to add this page to your Home for quick one-click access in the future.

Summary section (at the top of the page)

Notice [in the screen example below] that the name of the job step you are viewing is adjacent to the page title, Job Step Details.

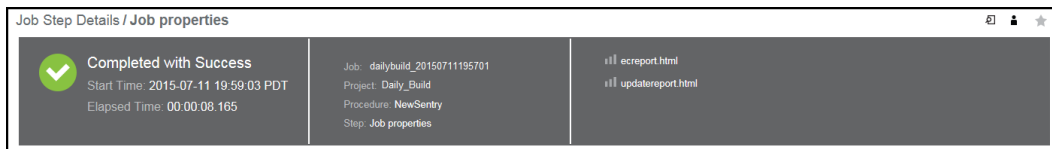
The Summary section provides:

- The job's Start, Wait (waiting for a resource), and Elapsed times
- You will see wait times specified if your job steps were restricted by resource, workspace, a precondition, or license availability.
For example, you will see the license wait time if a job step could not run because your license-allowed maximum concurrent step limit was met or exceeded, meaning no steps will run until the number of concurrent steps is reduced.

- A General Information section with links to get you to the job step's Job Details, Project Details, Procedure Details, or Edit Step pages.
- If the job created any custom reports, they appear in the summary area to the right of General Information. Reports on this page are for command steps only.

Reports

To create a special-purpose HTML report for a job: In the top-level of any of the job's workspaces, add a file that contains the word "report" or "Report" and ends in the `.html` file extension. The link is automatically created when the Job Details page is displayed. The job's workspace must be accessible to the web server. The link will be in the Reports section, within the Summary section.



Links

This page can include a Links section also, which will appear on the far right-side of the Summary section.

To add a link to this section:

- **Create a link to any file:**

To add a link, run a command in any job's job step using this format (this works for both local [disconnected] and non-local workspaces):

```
ectool setProperty "/myJobStep/report-urls/<myReportName>"
"/commander/jobSteps/<jobStepId>/<artifactDirectoryName>/<file-path>"
```

Example command:

```
ectool setProperty "/myJobStep/report-urls/Test Report" "/commander/jobSteps/
$[/myJobStep/jobStepId]/testreport/index.html"
```

Note: If you are using this format (from ElectricCommander 4.2 and earlier):

```
ectool setProperty "/myJob/report-urls/<myReportName>" "/commander/jobs/<jobId>/
<workspaceName>/<artifactDirectoryName>]/<file-path>"
```

Example command:

```
ectool setProperty "/myJob/report-urls/Test Report" "/commander/jobs/$[/myJob/job
bId]/
$[/myJobStep/workspaceName]/testreport/index.html"
```

It will continue to work only for non-local workspaces. We recommend, however, that you change the command to use the newer format.

- **Create a link to any directory:**

To add a link, run the following style command in any job's job step:

```
ectool setProperty "/myJobStep/report-urls/Main Job Workspace"
"file:///WinStor2/scratch/chron55build/$[/myJob/jobName]"
```

Note: Links to a directory automatically work in Internet Explorer, but if using Firefox, local links are disabled unless the default security policy is modified or an extension is used. Refer to http://kb.mozillazine.org/Links_to_local_pages_don%27t_work for more details.

The "tabbed" Sections

The next section of tabs allows you to select the type of information you want to see. You may see six or fewer tabs, depending on the type of job step you are viewing.

Child Steps table

Note: The Child Steps table is available for subprocedure and broadcast steps only.

- Click on a Step Name to go to that step's Job Step Details page.
- Click the Log "icon" to see that job step's log information.
- Status - This column provides the job's step status.
 - Aborted - The step or job was aborted.
 - Canceled - The step did not run because the job was aborted or a previous step was aborted.
 - Error - The command failed or `postp` detected an error in the output.
 - Running - The step is currently running or a step inside the subprocedure call is running.
 - Skipped - The step did not run because the run condition evaluated to "false".
 - Success - The step completed with no errors or warnings.
 - Waiting for Resource or Workspace - The step is waiting to run as soon as a resource or workspace is available.
 - Warning - `postp` detected a warning message.

Note: The Status field can be over-ridden by `postp` to include more specific information. `postp` information may be linked directly to diagnostic information—also available by selecting the Diagnostics tab.

- Click on a Resource name to go to the Edit Resource page.
- Click the **Edit** link in the Action column to go to the Edit Step page.

General table

This section displays basic information including the step ID, step name, create and end times, and whether the step invoked a subprocedure or ran a command.

General

Diagnostics

Properties

Notifiers

General

Step Id

abfb5a9e-2841-11e5-ab4e-0050568bb6ca

Step Name

Job properties

Create Time

2015-07-11 19:57:02 PDT

Elapsed Time

00:00:00.165

End Time

2015-07-11 19:59:11 PDT

Last Modified

2015-07-11 19:59:11 PDT

Last Modified By

project: Commander

Run Condition

1

Exclusive Mode

none

Release Mode

none

Resource Name

default

Diagnostics table

If the job step or child step (for subprocedure and broadcast steps) generated warning or error messages, you will see those messages here. This section contains one or more diagnostics, each of which is a portion of a step's log file that was identified as "interesting" by the step's postprocessor. Typically, each diagnostic relates to an error or warning detected by the postprocessor. All diagnostics for a particular job step appear together.

Diagnostics	
Exit Code:	0
Error Handling:	Abort Job
Time Limit:	300

Parameters table

This section displays parameter values supplied to the top-level procedure when the job step was invoked and has these columns:

- Name
- Value

Note: This table is available for steps with child steps only.

Child Steps	General	Diagnostics	Parameters	Properties	Notifiers
Name	Value				
branch	3.0				
cm	lonestar-cm				
dbType	oracle				
platform	windows				
resource	ec-win				
skipServerUnitTests	0				

Properties table

This section displays information computed and/or used by the job step while it is running. After the job step completes, these properties are normally read-only.

- Click on a Property Name to edit that property.
- If a "folder icon" precedes a property name, it denotes a Nested Property Sheet.
- This section also provides **Create Property**, **Create Nested Property**, and **Access Control** links to update or enter additional information for the next time this job step is run. For more information on properties, see the [Properties](#) Help topic.

Notifiers table

If you have designated certain persons or groups to be notified of details for specific job steps, this section displays who receives an email notification, the type of notification, any condition, and a description [if supplied]. If a notifier is available, you can click on its name to view details or make modifications.

Job Step Time and waitTime Properties Explained

- **start** - The time when this job step began executing.

When the ElectricFlow server assigns a resource to run the step, it issues a `<runcommand>` request containing the expanded command body to that resource's agent. The "start" time is when that `<runcommand>` is sent to the agent.

- **finish** - The time when this job step completed.

When a command finishes executing, the ElectricFlow agent issues a `<finishcommand>` request containing the exit code and other metadata back to the ElectricFlow server. The "finish" time is when the `<finishcommand>` is received by the server.

- **elapsedTime** - The number of milliseconds between the start and end times for the job.

This is exactly "finish" - "start".

- **totalWaitTime** - The sum of resource, workspace, and license wait times for this job step.

The sum of `resourceWaitTime`, `licenseWaitTime`, and `workspaceWaitTime`.

- **waitTime** - The number of milliseconds the step spent between runnable and running (for example, waiting for a resource).

Covers known reasons why a step couldn't run as soon as it became runnable (same as `totalWaitTime`), plus any other possible reason (slow transaction commits, system paging, etc).

- **resourceWaitTime** - The length of time this job step stalled because it could not get a resource. This could indicate that eligible resources for the step have reached their step limit, are in-use but the step requires a resource exclusively, or resources are down.

For subprocedure steps, this is the sum of all child steps.

- **licenseWaitTime** - The length of time this job step had to wait to process because the license limit was reached or exceeded.

For subprocedure steps, this is the sum of all child steps.

- **workspaceWaitTime** - The time this job step had to wait because no workspace was found or available.

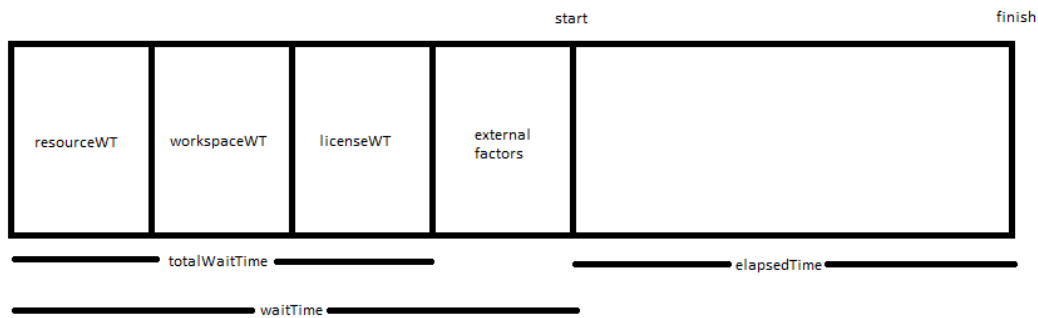
For subprocedure steps, this is the sum of all child steps.

When it is time to schedule the step, ElectricFlow checks the license and starts counting time against `licenseWaitTime` until the license is satisfied (e.g., running this step would not exceed the maximum number of licensed hosts). After the step is scheduled, ElectricFlow sends a `runCommand` message to the agent. If it fails because ElectricFlow cannot talk to the agent, ElectricFlow accrues time against `resourceWaitTime`. If ElectricFlow can talk to the agent, but it has trouble manipulating the workspace (for example, it cannot create the workspace dir, map the drive [on Windows], or create files), ElectricFlow accrues time against `workspaceWaitTime`.

It is possible for the following to occur:

1. A job step becomes stalls due to a license issue.
2. The license limit is no longer exceeded and ElectricFlow tries to run the step but hits a bad resource.
3. When the resource is good, the step stall because of a license again (adding more time to the licenseWaitTime),
4. When the license is satisfied, ElectricFlow reaches the resource, and everything runs.

This is why ElectricFlow continues cycling through these states until everything is satisfied.



The first 4 blocks (resourceWaitTime, workspaceWaitTime, licenseWaitTime, and external factors) can occur in any order.

External factors could include things such as system paging, slow transaction commits, etc.

Job Details

This page displays detailed information about a job. If the job is currently running, the page updates automatically as the job progresses. After the job is complete, the page will not continue to refresh. You can disable the automatic page refresh by clicking the **Stop Refresh** link at the top of the page.

Links and actions at the top of the table

- **Abort** - Aborts a running job.
- **Run Again** - Use this link to run the job again.
When you select the **Run Again** button, you are requesting to re-run the job with the same parameter values as the original invocation.
If you change the parameter values, you can select "**Run...**" from the dropdown menu (small down-arrow) next to the Run Again button.
- **Delete** - Deletes this job.
- **Save Configuration** - Saves this configuration to the Quick View section on the Home page.
- **Access Control** - Use this link to set privileges for this job. For more information, see the [Access Control](#) Help topic.
- **Stop Refresh** - This link is available only if a job is currently running. Click this link if you do not want the page to refresh automatically while a job is running.

- **View Log** - This link takes you to the Job Log page. If your job "completed with errors", this log contains error messages.
- The "star" icon allows you to save this job information to your Home page.
- The "curved arrow" icon returns you to the Jobs page.
- **Pagination** - Use the "previous" and "next" arrow icons to view the previous or next job. The numbers between the arrow icons display the number of jobs you can view and the first number indicates which job [in the list] you are viewing.

Summary section (at the top of the page)

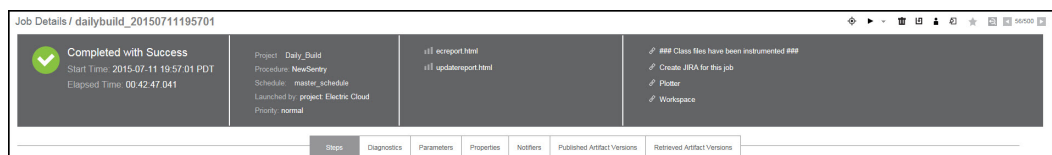
Along with the job's start and elapsed times, the summary section provides links to get you to the job's Project or Procedure Details pages or to the Edit Schedule page.

This section also displays:

- A job's priority and a "Calling State" category - If the job was called by a workflow, you will see two links, one to the Workflow Details page and one to the State Details page.
- "Wait times" - You will see *wait times* specified if your running job was restricted by resource, workspace, a precondition, or license availability.
Note: Total job wait time is more accurately thought of as *accumulated* wait time because it is the sum of all step wait times, including wait times for steps running in parallel.
- Two other useful sections: Reports and Links. If the job created any custom reports, they appear in the summary area to the right of **General Information**.

Reports

To create a special-purpose HTML report for a job: In the top-level of any of the job's workspaces, add a file that contains the word "report" or "Report" and ends in the .html file extension. The link is automatically created when the Job Details page is displayed. The job's workspace must be accessible to the web server. The link will be in the Reports section. The screen capture below displays an example of the HTML Reports section.



Click on a report file name to display the report.

Links

Notice the additional Links section in the screen example above. To add a link to this section:

- **Create a link to any file:**

To add a link, run a command in any job's job step using this format (this works for both local [disconnected] and non-local workspaces):

```
ectool setProperty "/myJobStep/report-urls/<myReportName>"
"/commander/jobSteps/<jobStepId>/<artifactDirectoryName>/<file-path>"
```

Example command:

```
ectool setProperty "/myJobStep/report-urls/Test Report" "/commander/jobSteps/
$[/myJobStep/jobStepId]/testreport/index.html"
```

Note: If you are using this format (from ElectricCommander 4.2 and earlier):

```
ectool setProperty "/myJob/report-urls/<myReportName>" "/commander/jobs/<jobId>/
<workspaceName>/<artifactDirectoryName>[/<file-path>]"
```

Example command:

```
ectool setProperty "/myJob/report-urls/Test Report" "/commander/jobs/$[/myJob/jo
bId]/
$[/myJobStep/workspaceName]/testreport/index.html"
```

It will continue to work only for non-local workspaces. We recommend, however, that you change the command to use the newer format.

- **Create a link to any directory:**

To add a link, run the following style command in any job's job step:

```
ectool setProperty "/myJobStep/report-urls/Main Job Workspace"
"file:///WinStor2/scratch/chron55build/$[/myJob/jobName]"
```

Note: Links to a directory automatically work in Internet Explorer, but if using Firefox, local links are disabled unless the default security policy is modified or an extension is used. Refer to http://kb.mozillazine.org/Links_to_local_pages_don%27t_work for more details.

The "tabbed" sections

The next section of tabs allows you to select the type of information you want to see.

Steps table

The Job Steps table displays all steps for this job.

View: All ▾							Expand All Collapse All	
Steps Diagnostics Parameters Properties Notifiers Published Artifact Versions Retrieved Artifact Versions								
Step Name	Log	Status	Elapsed Time	Resource	Actions			
CICheckout		Completed with Success	00:01:59.538					
PreCheckoutStep		Completed with Success	00:00:03.500	ecbuild- lin3				
CheckoutStep		Completed with Success	00:01:49.851					
checkoutMethod		Completed with Success	00:01:42.507					

- Mouse-over the **View: All** link [below the tabs] to see a filter list. You can choose a filter to apply to the Job Step table.
- The **Expand All | Collapse All** links are provided to see all subprocedures or to minimize the information presented.
- Click on a Step Name to go to the Job Step Details page for that step.
- Click the Log icon for any step to display the step's log file. This icon is not displayed if the `logFileName` property was removed from the step.

- **Status column:** This column provides the job's step status.
 - **Aborted** - The step or job was aborted.
 - **Canceled** - The step did not run because the job was aborted or a previous step was aborted.
 - **Error** - The command failed or `postp` detected an error in the output.
 - **Running** - The step is currently running or a step inside the subprocedure call is running.
 - **Skipped** - The step did not run because the run condition evaluated to "false".
 - **Success** - The step completed with no errors or warnings.
 - **Waiting for Resource or Workspace** - The step is waiting to run as soon as a resource or workspace is available.
 - **Warning** - `postp` detected a warning message.

Note: The Status field can be over-ridden by `postp` to include more specific information. `postp` information may be linked directly to diagnostic information—also available by selecting the Diagnostics tab.

Action column:

- Click the **Edit** link to edit the procedure step definition.
- Aborting a step requires the execute privilege on the job—not just the modify privilege.

When you click **Abort**, the dialog box similar to the following appears:



Click **OK** to abort the step. If the step does not complete in approximately two minutes, you can "forcibly" abort the step to force the step to a completed status, enabling the rest of the job to continue.

Click **Cancel** if you change your mind.

Diagnostics table

If the job generated warning or error messages, click this tab to see one or more diagnostics, each of which is a portion of a step's log file identified as "interesting" by the step's postprocessor. Typically, each diagnostic relates to an error or warning detected by the postprocessor. All diagnostics for a particular step appear together.

Each diagnostic contains a header line such as "Error #3 of 5" with an icon, followed by a line with additional identifying information and one or more lines extracted from the log file. The identifying line has a format similar to:

```
"systemTest-windows.log:16 (systemtest) "
```

- The first part of the line provides the name of the log file (`systemTest-Windows.log`) and the line number of the first line in the file that is part of this diagnostic (16).
- If you click on the log file name, ElectricFlow displays the full log file for you, with the portion in this diagnostic highlighted. If the postprocessor was able to provide additional information about the issue, it is displayed in parentheses after the log filename. The first value (`systemtest-windows`) gives the name of a particular module or package in which the problem occurred.
- The second value (`systemtest`) identifies a specific place where the problem occurred, such as the file name that did not compile or the name of a test that failed.

Parameters table

This section displays parameter values supplied to the top-level procedure when the job was invoked. This table does not allow you to edit the parameter's value.

These parameters are copied automatically to the top-level property sheet and you will see them in the properties table also.














Parameters	
Name	Value
branch	main
generateReport	1
mavenDebug	0
preFlight	0
preFlightTag	
preFlightUser	

Properties table

This table contains information computed and/or used by the job while it is running.

- All defined parameters are automatically copied to the top-level property sheet. Any custom properties defined for the state will be included in this table also.
- After the job completes, these properties are normally read-only.
- Click on a Property Name to edit that property.
- If a "folder icon" precedes a property name, it denotes a Nested Property Sheet.
- If you use square brackets in a property name, the property must be encapsulated in square brackets to avoid property sheet expansion. Note that this works for specific functions, such as creating a link on a job, but the drawback is that the property cannot be edited or deleted from the web interface. Example:

```
ectool setProperty "/myJob/report-urls[This property name has [some square bracket] text in it]" "http://website.com"
```
- This section also provides **Create Property**, **Create Nested Sheet**, and **Access Control** links to update or enter additional information for the next time this job is run.

Steps Diagnostics Parameters Properties Notifiers Published Artifact Versions Retrieved Artifact Versions						
					Create Property Create Nested Sheet Access Control	
Property Name	Value				Description	Actions
 ecscm_changeLogs						 
 ecscm_snapshots						 
 report-urls						 
P4CLIENT	cmdr-imp-client-ae333861-2841-11e5-ab4a-005056bb6bca					
branch	main					
buildNumber	94404					
generateReport	1					

Notifiers table

If you designated certain persons or groups to be notified of details for specific jobs, this section displays who receives an email notification, the type of notification, any condition, and a description [if supplied]. If a notifier is available, you can click on its name to view details or make modifications.

Steps

Diagnostics

Parameters

Properties

Notifiers

Published Artifact Versions

Retrieved Artifact Versions

Currently, there are no records to display in this list.

Published Artifact Versions table

This table displays artifact versions published by steps in this job.

Steps

Diagnostics

Parameters

Properties

Notifiers

Published Artifact Versions

Retrieved Artifact Versions

Name	State	Description	Actions
AM:Deploy:1.0.0-1	available		Edit Delete

Column descriptions

- **Name** - Select this name to go to the Artifact Version Details page for this artifact version.
- **State** - Possible values for the state of the artifact version are: `available|publishing|unavailable`
- **Description** - A previously supplied text or HTML description for this artifact version.
- **Actions column:**
 - **Edit** - Select this link to go to the Edit Artifact Versions page.
 - **Delete** - Select this link to delete the artifact version. You may be prohibited from deleting this artifact version if you do not have sufficient access control permissions.

Retrieved Artifact Versions table

This table displays artifact versions retrieved by steps in this job.

					Steps	Diagnostics	Parameters	Properties	Notifiers	Published Artifact Versions	Retrieved Artifact Versions
Name	State	Description									
ARM-Deploy1.0.0-1	available										
ARM-Deploy1.0.0-1	available										

Column descriptions

- **Name** - Select this name to go to the Artifact Version Details page for this artifact version.
- **State** - Possible values for the state of the artifact version are: `available|publishing|unavailable`
- **Description** - A previously supplied text or HTML description for this artifact version.
- **Actions column:**
 - **Edit** - Select this link to go to the Edit Artifact Versions page.
 - **Delete** - Select this link to delete the artifact version. You may be prohibited from deleting this artifact version if you do not have sufficient access control permissions.

Licenses

This page displays all license information known to the ElectricFlow server. Typically, a single license is displayed, which describes the usage to which you are entitled. The license on the server may be based on concurrent resources, concurrent users, concurrent steps, registered hosts, or any combination of these licenses.

All Licenses

Click the **Import License** link to import a new license file.

Column descriptions

Column Name	Description / Actions
Company	The name of the company or organization to which the license was granted. If this is not your company, please contact Electric Cloud Customer Support.
Feature	The type of license.
Expiration	The last day when this license is no longer valid. After your license expires (and after any grace period), running jobs will complete, but no new jobs can be launched.
Grace Period (days)	If this field is nonzero, you may continue using ElectricFlow for this many days after the expiration date. However, you will be warned frequently during the grace period that your license has expired.
Actions	View License - use this link to see additional details about this license. Delete - click this link to delete this license.

Current Usage

You see some or all of the following fields depending on the type of license or licenses you use for ElectricFlow.

Field Name	Description
Maximum Concurrent Steps	The maximum number of concurrent steps allowed by this license.
Concurrent Steps	The number of concurrent steps running now.
Maximum Managed Hosts In Use	The maximum number of unique host names defined in <i>managed</i> resources (non-proxy resources), where steps can run at any time.
Managed Hosts in Use	The number of unique host names defined in <i>managed</i> resources where steps are currently running.
Maximum Proxied Hosts In Use	The maximum number of unique host names defined in <i>proxied</i> resources, where steps can run at any time.
Maximum Concurrent Hosts	The maximum number of unique host names allowed to use ElectricFlow concurrently at any given time.

Field Name	Description
Concurrent Hosts in Use	The number of unique host names defined in concurrent resources where steps are currently running.
Maximum Proxied Host	The maximum number of unique host names define in proxied resources.
Proxied Hosts in Use	The number of unique host names defined in <i>proxied</i> resources where steps are currently running.
The following fields are specific to a concurrent user-based license.	
Maximum Concurrent Users	The number of users allowed to use ElectricFlow concurrently at any given time.
Active Users	The number of users with an active license.
User IP	The "user@IP" combination that holds an active license.
Expiration	The time when the license will be released.
Last Use	The time when the license was last used.
The following fields are specific to a registered host-based license.	
Maximum Registered Hosts	The maximum number of registered hosts allowed by this license.
Registered Hosts in Use	The number of registered hosts where steps are currently running.

License Types

Concurrent Resource License

This ElectricFlow license controls the number of unique host names on which steps can be scheduled at any time. Two properties, `maxHosts` and `maxProxiedHosts`, are defined on a license to control the maximum number of managed and proxied hosts that can be in use at the same time. The host named "localhost" and "127.0.0.1" refer to the local agent installed on the ElectricFlow server machine, and these host names are not part of the total count for hosts in use.

If two or more steps are running concurrently on the same host, the host name still counts once only towards the license limit. There is no limit to the number of resources pointing to a single host; from a licensing perspective, they all count as a single host. When the host limit is met, any new job step that attempts to run on a host [not already in use] will stay in the "runnable" state until its host is available. Current license usage statistics are

displayed at the top of the Resources page in the web interface. If no license is available, jobs cannot be launched. When a license expires, running jobs are allowed to complete, but no new jobs can be launched.

Concurrent User License

If the `maxConcurrentUsers` property is set on the license, its value defines the maximum number of users who can actively use the ElectricFlow server. A unique license is defined by the combination of the user's name and the IP address from which the user is making requests. A user can have multiple active sessions (through the web UI and ectool) from the same machine while only using a single license.

If a user makes a licensed API call without an active license, the server automatically attempts to assign a license. If a license is available (because the maximum has not been met), the user is assigned a license. If no licenses are available, the user will see a `NoAvailableLicenses` error and be redirected to the licenses page. Currently held licenses and their expiration dates can be viewed by calling the `getLicenseUsage` API or by viewing the Active Users table on the Licenses web page. When a license becomes available, the first user to make a licensed API call will be assigned that license. The only API calls not licensed are `login`, `getLicenseUsage`, `getAdminLicense`, `getServerStatus`, and `getServerInfo`.

When a license is first acquired, its expiration is set to 1 hour in the future. After the initial hour has passed, each time the user makes a licensed API call, the license is extended another 10 minutes (a configurable server setting). When a user's license passes the expiration date, it is released. When the user makes another licensed API call, a license is assigned automatically if one is available, otherwise the `NoAvailableLicenses` error is displayed.

A single administrator license can be used in an emergency. All users who have `execute` permission on the system-level licensing access control list can use the administrator license by calling the `getAdminLicense` API. This license has no expiration as long as all regular licenses are in use, but this license abides by the regular license expiration policy if regular licenses are available.

Concurrent Step License

This ElectricFlow license controls the number of unique steps you are allowed to run in the system at the same time (in parallel). The `maxConcurrentSteps` property, defined on a license, controls the maximum number of steps that can be running at the same time.

Registered Host License

You can use registered hosts for ElectricFlow. Registered hosts are dedicated hosts on which a procedure or step can always run. This license controls the number of unique registered host names. Two properties, `maxHosts` and `maxProxiedHosts`, are defined on a license to control the maximum number of registered and proxied hosts that can be in use at the same time. The host named "localhost" and "127.0.0.1" refer to the local agent installed on the ElectricFlow server machine, and these host names are not part of the total count for hosts in use. Concurrent hosts cannot be used with registered hosts.

If two or more steps are running concurrently on the same host, the host name still counts once only towards the license limit. There is no limit to the number of resources pointing to a single host; from a licensing perspective, they all count as a single host. When the host limit is met, any new job step that attempts to run on a host (not already in use) stays in the "runnable" state until its host is available. Current license usage statistics are

displayed at the top of the Resources page in the web interface. If no license is available, jobs cannot be launched. When a license expires, running jobs are allowed to complete, but no new jobs can be launched.

View License

This page displays all available license details for a single license.

Use the **Delete** link to remove this license.

Information

Name	Description
Company Name	The name of the company or organization to which the license was granted. If this is not your company, contact Electric Cloud Customer Support.
Product Name	Product to which this license applies (if it is not "ElectricFlow," it will not apply here).
Feature Name	Product feature to which this license applies: must be "Server" for ElectricFlow licenses.
Maximum Concurrent Steps	The maximum number of concurrent steps allowed by the license.
Maximum Managed Hosts In Use	The maximum number of unique host names defined in managed resources (non-proxy resources), where steps can run at any time.
Maximum Proxied Hosts In Use	This is the maximum number of unique host names defined in proxied resources, where steps can run at any time.
Maximum Registered Hosts	The number of registered hosts on which a procedure or step can always run. This appears when the license on the ElectricFlow server is based on a combination of concurrent and registered hosts or only registered hosts.
Maximum Concurrent Users	The number of users allowed to use ElectricFlow concurrently at any given time.
Expiration Date	The last day when this license is no longer valid. After your license expires (and after any grace period), running jobs will complete, but no new jobs can be launched.

Name	Description
Grace Period	If this field is non-zero, you may continue using ElectricFlow for this many days after the expiration date. However, you will be warned frequently during the grace period that your license has expired.

Import License

Use this page to provide a new license for ElectricFlow.

You should have received a license (as a block of text) from Electric Cloud already. If you have not received a license, contact Electric Cloud Customer Support or your sales representative.

- Use a text editor to open the license file.
- Copy the text into the Contents field.
- Click **OK**.

You will see your license information on the Licenses page.

Plugin Manager

A plugin is a collection of one or more features that can be added to ElectricFlow.

- Plugins are delivered as a JAR file containing the features implementation.
- When a plugin is installed, the ElectricFlow server extracts the JAR contents to disk into a configurable plugins directory.
- A plugin has an associated project that can contain procedures and properties required by the implementation.
- A plugin can provide one or more new pages for the web interface and may also provide a configuration page so you can provide any additional information that may be necessary to implement the plugin.

ElectricFlow provides default/bundled plugins, installed during the ElectricFlow installation. These plugins are "authored" and generally supported by Electric Cloud.

The information Plugin Manager displays across the top of the table:

Tab and field descriptions above the table

Currently Installed - The Plugin Manager default page view displays currently installed plugins. If you install additional ElectricFlow plugins from other sources, including those plugins you may create, they will also appear on this page.

Install from File/URL - Select this tab to see the following screen for plugin installation options:

File Install - Use this method to upload and install a local "jar" file from the machine running your web browser. Click **Upload** after specifying the path you need.

URL Install - install a plugin from a location specified by an URL. This location can either be an external web server (using `http://`), or a file on the ElectricFlow server host (using `file://`)

Install a plugin from the command-line:

ectool - the command-line tool, contains a full set of commands to perform plugin tasks. For more information on available ectool plugin commands, see the "API Commands - Plugin Management" topic in the ElectricFlow API Guide.

Note: The server has a configurable maximum file upload size that defaults to 50MB. Refer to the server "Maximum upload size" setting to adjust the default.

View Catalog - This page displays all plugins available for installation. If you do not see an Install option in the Actions column, you may not have the necessary access control permissions to install a plugin.

Catalog Install - click on the **Install** action (next to a plugin listed in the catalog) to download and install the plugin.

Category: a drop-down menu to filter the plugin list by a specific category.

Column descriptions

Column Name	Description / Actions
Plugin Label	The name of the plugin. Click on the label of an installed plugin to go to the Project Details page for that plugin.
Author	Identifies the person or organization that created the plugin and optionally provides a link to contact the author.
Version	The plugin version. Promoted plugin versions are also shown in this column. A promoted version number is labeled "(promoted)" immediately after the version number.
Category	The plugin type. Plugins that provide similar types of functionality are grouped into categories.
Description	A short text string that describes the plugin purpose or function.

Column Name	Description / Actions
Actions	<p>This column contains one or more of the following links to perform plugin actions or tasks. Not all plugins have the same task options. For example, some plugins may perform their own "setup" during installation and not require additional configuration information.</p> <p>Note: If you do not see the option you need in the Action column, it may be hidden from your view if you do not have sufficient access control permissions.</p> <p>Possible Actions:</p> <p>Configure - use this link to go to the configuration page for the plugin.</p> <p>Demote - use this link to make this plugin inactive and remove any tabs associated with the plugin.</p> <p>Note: If you re-install a previously demoted plugin, previous values from the demoted version are not copied to the new version you install.</p> <p>Help - if provided, use this link to go to the plugin documentation.</p> <p>Install - Download and install a plugin from the plugins catalog site.</p> <p>Promote - use this link to upgrade the plugin to a new version. Old plugin values are copied to the promoted version.</p> <p>Re-install - use this link to download and reinstall a previously installed plugin.</p> <p>Uninstall - use this link to remove this plugin from your system.</p> <p>Note: the Demote, Promote, Install, Re-install, and Uninstall actions appear only if the user has modify permission to the "plugins" ACL.</p>

Using a plugin

Multiple methods for adding plugin functionality to ElectricFlow are available.

- To configure ElectricFlow to recognize your SCM plugin, see the [Configuring ElectricFlow](#) Help topic and select the *Setting Up a Source Control Configuration* section.
 - For additional plugin Help, see the [Customizing the ElectricFlow UI](#) Help topic.
 - For additional help using your SCM plugin with ElectricFlow Preflight, Default Tracking, and ElectricSentry, see those Help topics: [Preflight Builds](#), [Defect Tracking](#), [ElectricSentry](#)
- To add a subprocedure step using the Edit Step or New Step page in the ElectricFlow UI:
 1. Select the Projects tab and choose an existing project or click the **Create Project** link if you need to create a new project.

2. On the Project Details page, select an existing procedure or click the **Create Procedure** link if you need to create additional procedures.
3. On the Procedure Details page, to create a New Step using a plugin, select the **Plugins** link to go to the Choose Step panel.
 - In the left pane, select a plugin category and then choose a plugin.
 - In the right pane, select the step type you want, which will take you to the New Step page to add a new subprocedure step (using your selected plugin).
4. On the New Step page, notice that the plugin you selected is already displayed in the Subprocedure section.
5. Enter information in the Parameters section fields. These fields are specific to the plugin you chose. (These fields are frequently different for each plugin.)

Configuring the plugins directory

For instructions on configuring the plugins directory for your ElectricFlow server, remote agents, and/or remote web servers, see the *ElectricFlow Installation Guide*.

Procedure Details

This page displays procedure component tables, including a table for Procedure Steps, Parameters, Email Notifications, and Custom Procedure Properties. A procedure describes a process to execute that consists of one or more step.

Use this page to create steps.

Procedure Steps

This table displays all steps in the procedure [named at the top of the page]. The steps execute in order from top to bottom. Normally only one step executes at a time, but if a group of adjacent steps is marked as "parallel," all of those steps will execute simultaneously.

Important: If you want to change the order of the steps, click on the icon to the left of the step name and drag it up or down to the new step position.

Note: This table displays only a few of the fields for each step. To view (and modify) the complete details for a step, click on the step's name to go to the Edit Step page.

Links at the top of the page

- Use the **Run** link [hovering your mouse over the small down-arrow on the right-side of the link] to choose Run Immediately or Run...
 - Run Immediately - Selecting this option runs the procedure immediately as set.
 - Run... - Selecting this option displays the Run Procedure web page where you may alter any existing parameters for this procedure, or set an existing credential.
- Click the **Access Control** link to go to the Access Control page to add privileges for this procedure.
- Click the **Edit** link to go to the Edit Procedure page so you can make modifications if necessary.

- Use the **Copy** link in the Actions column to make an exact copy of this step.
- Choose one of the New Step links to create the type of step you need.

Creating a new step

The following list describes some of the more common steps you can create.

- **Command** - this link takes you directly to the New Step page to write your own step. A Command(s) text box is available to add a script. This step invokes a `bat`, `cmd`, `shell`, `perl` script, or similar.
- **Subprocedure** - the Choose Subprocedure Step dialog is displayed, allowing you to select a project or plugin from which you want to call an existing procedure to create a step. This step invokes another ElectricFlow procedure and will not complete until all subprocedure steps have finished.
- **Plugin** - this link opens the Choose Step panel. The left pane displays plugin categories and the right pane displays available pre-configured steps for your left-pane selection. To quickly find your application/plugin, type your selection name in the Search box. Select the step you want to create and ElectricFlow takes you to the New Steps page.
 - On the New Step page, notice that the plugin step you selected is already displayed in the Subprocedure section.
 - Enter information in the Parameters section fields. These fields are specific to the plugin you chose. (These fields are different for each plugin.)

Enter information in any remaining fields to complete your step. See the New Step Help topic for more information.

A note about plugins

Plugins are the vehicle ElectricFlow uses to integrate source control systems (such as Subversion or Perforce), defect tracking applications (such as JIRA or MKS), reporting functionality, code analysis applications, build applications, and much more. To see the complete list of plugins bundled and installed with ElectricFlow, see the Plugin Manager page (go to Administration > Plugins).

Note: If you do not see the SCM system or defect tracking system you prefer, a Plugin Catalog is available from the Plugin Manager also. All plugins in the catalog are available for installation into ElectricFlow.

Parameters

Each procedure can define parameters, which are values provided to the procedure when it is invoked. Each parameter value is placed in a property associated with the job and the procedure can use the property to control its execution. For example, a parameter might specify a particular branch of a product to build; another parameter might indicate whether unit tests should be run during this build.

When you define a parameter, indicate whether the parameter is required (meaning the procedure will not run unless a value is provided for the parameter). Also, you can provide a default parameter value and a description to help callers understand how to use the parameter.

- Click the **Create Parameter** link to go to the New Parameter page.
- Click on the Parameter Name to edit its contents.

Email Notifiers

Use the **Create On Start Email Notifier** or **Create On Completion Email Notifier** links to go to one of the respective pages to create an email notifier for this procedure.

After creating an email notifier, click on the "bread crumb" link at the top of the page to return to the Procedure Details page.

Custom Procedure Properties

- To add new properties to this procedure, click the **Create Property**, **Created Nested Sheet**, or **Access Control** links.
- Click on the Property Name to edit its contents.
- Nested Property Sheets appear in this table preceded by a folder icon.

Tip

Want to rename this procedure or change its description? Click the **Edit** link at the top of the page.

Procedure - create new or edit existing procedure

To create a new procedure

Enter information in the fields as follows:

Field Name	Description
Name	Enter a name for the procedure. This name must be unique within the project.
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code>

Field Name	Description
Job Name Template	<p>This is the template used to determine the default name of jobs launched from the procedure.</p> <p>For example:</p> <pre> \$[projectName]_\$_[/increment /myproject/jobCounter]_ \$_[/timestamp] </pre> <p>Produces a name like:</p> <pre> projectFoo_1234_20140102130321 </pre> <p>Enter any combination of elements to create procedure names more meaningful to you. For example, you could choose to include the build number.</p> <p>Note: Electric Cloud does not recommend using the ElectricFlow-generated <code>jobId</code> because it is no longer a human-readable integer so it does not provide any indentifiable information and cannot be used as a counter.</p>
Default Resource	Enter a resource name. This name must be unique among other resource names or click Browse to select a resource.
Default Workspace	Enter the name of the workspace or click Browse to select a workspace name.
Time Limit	<p>If no time limit was specified on the calling step, time limits are copied to the calling step from the procedure. If the procedure is called from <code>runProcedure</code> (or a schedule), the time limit acts as a global job timeout.</p> <p>The "timer" for the procedure starts as soon as the calling step/job becomes runnable (all preconditions are satisfied).</p> <p>Specify a number and then use the pull-down menu to select the "time units" of the number you specified.</p>
<i>Impersonation Credential</i>	
Credential Project	Default Credential Project is the current project, or select the other button and Browse for the project name you need.
Credential Name	The name of the credential for the project - you may need to Browse to find this name.

Click **OK** to save the information to create a new procedure.

To edit an existing procedure

Use this page to modify certain overall information about a procedure. Additional information, such as procedure steps and properties, can be viewed and modified on the Procedure Details page. To access this page, click the **Project Name** link on the location path (breadcrumb), then select a Procedure name.

To learn more about a particular field on the form, mouse-over the field to pop-up descriptive text under the mouse or click the **Help** link at the top of the page.

Tip

Want to rename the procedure? Enter a new procedure name in the Name field, click **OK**.

Step - create new or edit existing step



Use this page to define a new step or to modify an existing step. The following information describes the common fields you need to enter to create any type of step.

Note: If you chose to create a step using the Plugin link, additional help is available in the Parameters section by clicking the "?" icon. The Parameter section is populated according to type you chose to create. Different step types require different information.

To create a new command or subprocedure step

Enter information in the fields as follows:

Field Name	Description
<i>General section (for all step types)</i>	
Name	Enter a name for the step that must be a unique name within the procedure.
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code>
<i>Command section (for command steps only)</i>	
Command(s)	A script to execute the step functions that is passed to the step's shell for execution.

Field Name	Description
Resource	<p>Enter a unique resource or resource pool name or click Browse to select a resource. If you leave this field blank, the procedure or project provides a default resource.</p> <ul style="list-style-type: none"> • The job step must have the execute privilege on the resource or resource pool. If it does not, the job step will be canceled with an access denied error. • The resource or resource pool must be enabled. If the resource or pool is not enabled, no error is generated, and the step remains in the runnable state until the pool is enabled.
Postprocessor	<p>If this field remains blank, no postprocessor runs for the step. If this field is not blank, it specifies a command (passed to the step's shell for execution) that analyzes the log file for the step and collects diagnostic information for reporting. <i>For more information</i>, see the Postprocessors Help topic.</p>
<i>Subprocedure section (for subprocedure steps only)</i>	
Procedure	<p>Displays the current project and procedure. Click Change to use the pop-up menu to select a new project or procedure.</p>
Resource	<p>Enter a unique resource name or click Browse to select a resource. If you leave this field blank, the procedure or project provides a default resource.</p>
<p><i>Parameters section (for subprocedure steps only)</i></p> <p>This section expands automatically, entering the fields you need specifically for the Subprocedure you chose in the previous section. For additional help with field descriptions, click the "?" icon if available.</p> <p> To toggle between the standard view and property view for this section, click the  icon. This icon appears in the standard form when one or more of the parameter types are not Text entry or Text area. It appears in the custom form for all the supported parameter types when the form has at least one parameter.</p>	
<i>Advanced section (for both command and subprocedure steps)</i>	

Field Name	Description
Precondition	<p>By default, if this field is blank, the step has no precondition and will run when scheduled. Set this property to make a step wait until one or more dependent conditions are met. When a job step is eligible to transition from pending to runnable, a <i>precondition</i> is evaluated.</p> <p>A <i>precondition</i> is a fixed text or text embedding property reference that is evaluated to TRUE or FALSE. An empty string, a \"0\" or \"false\" is interpreted as FALSE. Any other result string is interpreted as TRUE. The step will block until the precondition is TRUE.</p> <p>Precondition example: Assume we defined these 4 steps:</p> <ol style="list-style-type: none"> 1. Build object files and executables 2. Build installer 3. Run unit tests 4. Install bits on test system <p>Step 1 is an ordinary serial step. Steps 2 and 3 can run in parallel because they depend only on step 1's completion. Step 4 depends on step 2, but not step 3.</p> <p>You can achieve optimal step execution order with preconditions:</p> <ul style="list-style-type: none"> • Make steps 2-4 run in parallel. • Step 2 needs a job property set at the end of its step to indicate step 2 is completing (/myJob/buildInstallerCompleted=1). • Set a precondition in step 4: \$[/myJob/buildInstallerCompleted]
Run Condition	<p>By default, if this field is blank the step will always run. This field accepts "fixed text" or text embedding property references that are evaluated into a logical TRUE or FALSE. A "0" or "false" is interpreted as FALSE. An empty string or any other result is interpreted as TRUE.</p> <p>The property reference can be a JavaScript expression, for example, this expression could test whether the name of a step is equal to the value of a property called "restartStep". \$[/javascript (myStep.stepName == myJob.restartStep)]</p>

Field Name	Description
Error handling	<p>This field determines what happens to the procedure if the step fails.</p> <p>The first choice, "procedure continues, but overall status will be error" means an error in the step does not abort the procedure; subsequent steps will run as usual. However, an error will be reflected in the overall outcome of the procedure. If this is the top-level procedure in the job, the job will have an error outcome.</p> <p>If this is a subprocedure, the step invoking the subprocedure will have an error outcome, which still could cause the job to abort, depending on the error handling for that step.</p> <p>The second choice for this field is "abort procedure but allow running steps to continue." In this case, if an error occurs, no new steps are initiated in this procedure except those where "always run" was selected. Any steps currently running are allowed to complete, then the procedure will abort and its outcome will be set to error as described previously.</p> <p>Six error handling options are:</p> <ul style="list-style-type: none"> • <code>failProcedure</code> - The current procedure continues, but the overall status is error (default). • <code>abortProcedure</code> - Aborts the current procedure, but allows already-running steps in the current procedure to complete. • <code>abortProcedureNow</code> - Aborts the current procedure and terminates running steps in the current procedure. • <code>abortJob</code> - Aborts the entire job, terminates running steps, but allows <code>alwaysRun</code> steps to run. • <code>abortJobNow</code> - Aborts the entire job and terminates all running steps, including <code>alwaysRun</code> steps. • <code>ignore</code> - Continues as if the step succeeded.
Time Limit	<p>The maximum amount of time the step can execute. If the step exceeds this time, it will be aborted. Specify a number and then use the pull-down menu to select the "time units" of the number you specified.</p>
Run in parallel	<p>If this box is "checked," this step will run concurrently with any adjacent steps marked as parallel. If this box is not checked, the step will NOT run until all previous steps in the procedure have completed, and it will complete before any following steps execute.</p>

Field Name	Description
Always run step	If a procedure is aborted (either because of an error in a step or because the job was aborted), under normal conditions no new steps will start in the procedure. However, if this box is "checked," the step will run even if the procedure is aborted. This facility is most commonly used for steps at the end of a job that generate reports or perform cleanup operations. Note: Remember that the Run Condition field is still evaluated and may cause the step to be skipped even if this box is checked.
Retain Exclusive	Select one of the following options: <ul style="list-style-type: none"> • None - the "default", which does not retain a resource. • Job - keeps the resource for the duration of the job. No other job can use this resource, regardless of its step limit, until this job completes or "Release Exclusive" is used in a step. Future steps for this job will use this resource in preference to other resources—if this resource meets the needs of the steps and its step limit is not exceeded. • Step - keeps the resource for the duration of the step. • Call - keeps the resource for the duration of the procedure that called this step, which is equivalent to 'job' for top level steps.
Workspace	Use this field to define a workspace for the step to use. You can either type-in a workspace name or click Browse to select a workspace. If you leave this field blank, the workspace is determined by the procedure or project.
<i>These "Advanced" fields are used for command steps only</i>	
Broadcast	"Checking" this box means replicate the step to execute (in parallel) on each of the specified resources (for example, if a resource pool is specified, run the step on each resource in the pool).
Release Exclusive	Select one of the following options: <ul style="list-style-type: none"> • none - the "default" - no action if the resource was not previously marked as "retain". • release - releases the resource at the end of this step. If the resource for the step was previously acquired with "Retain exclusive" (either by this step or some preceding step), the resource exclusivity is canceled at the end of this step. The resource is released in the normal way so it may be acquired by other jobs. • releasetojob - allows a step to promote a Step exclusive resource to a Job exclusive resource.

Field Name	Description
Shell	<p>This shell will be used to execute the step's commands on a resource. For example, using <code>sh</code> or <code>cmd /c</code>, the agent saves the command block to a temporary file with a <code>.cmd</code> extension and runs it:</p> <pre>sh foo.cmd or cmd /c foo.cmd</pre> <p>If you do not specify a shell on a step, at step run-time the server looks at the resource shell. If a resource shell is not set, the shell line used by the agent is platform dependent:</p> <p>Windows: <code>cmd /q /c "{0}.cmd"</code> UNIX: <code>sh -e "{0}.cmd"</code></p> <p>When you specify a shell (in the step or resource), and omit the <code>cmd-file</code> marker, the agent notices the omission and takes the correct action. For example: a user specifies <code>sh -x</code>. The agent converts this to <code>sh -x "{0}.cmd"</code></p> <p>Two alternate forms of shell syntax where ElectricFlow uses a "marker," <code>{0}</code>, as a placeholder for the command file argument:</p> <ul style="list-style-type: none"> • <myShell> {0} <potential extra shell args> In this example, the command file is not meant to be the last argument in the final command line. For example, <code>mysql -e "source {0}"</code> This shell example runs the <code>mysql</code> command against this step's command containing <code>sql</code>. • <myShell> {0} <.file extension> <potential extra shell args> In this example, the shell requires the command file to end in an extension other than <code>.cmd</code>. For example, <code>powershell "& '{0}.ps1'"</code> This shell example runs Microsoft PowerShell against this step's command containing PowerShell commands. <p>Notes:</p> <ul style="list-style-type: none"> • When the agent parses the shell, it will parse the extension as everything after <code>{0}</code> . until it sees a space or non-alphanumeric character. • If your script uses International characters (non-ascii), add the following block to the top of your <code>ec-perl</code> command block: <pre>use utf8; ElectricCommander::initEncodings</pre>

Field Name	Description
Working Directory	The directory where the commands for this step execute. A relative name is interpreted relative to the root directory for the job workspace. If you leave this field blank, the step's working directory will be the top-level directory in the job workspace. Note: If this step will run on a proxy resource, this directory must exist on the proxy target. The step will run in this directory on the proxy target.
Log File	The step's log file name, specified relative to the root directory in the job workspace. If you leave this field blank, ElectricFlow picks a unique name for the log file based on the step name.
<i>Impersonation section (for both command and subprocedure steps)</i>	
Credential Project	By default, the Current project is used, or click Browse and select a different project.
Credential Name	If you select a credential in this field, the step runs under the account from that credential. This field is usually blank, which means a default credential is used. For more detailed information about credentials, see the Credentials and User Impersonation Help topic.

Click **OK** to create the step.

To edit an existing command or subprocedure step

You can modify any of your previously entered step information and add information to one or all of the four additional sections.

Attached Credentials

Click the **Attach Credential** link to retrieve an existing credential for this step. If you need to create a new attached credential, return to the Project Details page for this step.

Attached Parameter Credentials

Click the **Attach Parameter Credential** link to retrieve an existing parameter credential for this step.

Email Notifiers

Choose **Create On Start Email Notifier** or **Create On Completion Email Notifier** to go the respective page to create the email notifier you need.

Custom Step Properties

Choose from the **Create Property**, **Create Nested Property**, or **Access Control** links to add a custom property to this step. Each Property pop-up box has its own Help topic or go to the main [Properties](#) Help

topic for more information. For more information about Access Control, go to the main [Access Control](#) Help topic.

Click **OK** to save your modified step information.

Publish Artifact Version Step

Use the EC-Artifact plugin to create a step to publish a new artifact version.

The following parameters are available to create this step:

Field Name	Description
Artifact	Artifact name, in the form <groupId>:<artifactKey>. If the artifact does not exist, it will be created if this procedure's launching user or this project principal has the required permissions.
Version	A full version string takes the form: <major>.<minor>.<patch>-<qualifier>-<buildNumber>. The version specification must be unique across all of this artifact's versions.
Repository	Name of the repository where this new artifact version will be published.
Enable Compression	Check this box to compress the artifact version before it is stored in the repository.
From Directory	Name of the directory in the job's workspace containing files that comprise the artifact version to be published. If not specified, the entire workspace is used.
Include Patterns	List file "include" patterns one pattern per line, to limit which files are published. If no patterns are specified, all files are included.
Exclude Patterns	List file "exclude" patterns one pattern per line, to limit which files are published. If no patterns are specified, no files are excluded.
Dependent Artifact Versions	List dependent artifact versions one per line, each in the form <groupId>:<artifactKey>:<versionRange>. All dependent artifact versions must exist for this artifact version to be retrievable. When this artifact version is successfully retrieved, its dependent artifact versions are retrieved also.

Retrieve Artifact Version Step

Use the EC-Artifact plugin to create a step to retrieve an artifact version.

The following parameters are available to create this step:

Field Name	Description
Artifact	Artifact name, in the form <groupId>:<artifactKey>.
Version	Select the latest version (Latest), the exact version of the artifact (Exact), or a version range (Range). Version is in the form: <major>.<minor>.<patch>-<qualifier>-<buildNumber>
Retrieve to directory	If you want to retrieve this artifact version to a specific directory location, select the check box, then specify the full path to the directory where you want to retrieve this artifact version.
Overwrite	Use the drop-down menu to select one of the following options: <ul style="list-style-type: none">• <code>true</code> - deletes previous content in the directory and replaces the content with your new version.• <code>false</code> - (existing behavior) if the directory does not exist, one will be created and filled with the artifact's content. If the directory exists, a new directory is created with a unique name and the artifact contents is supplied there.• <code>update</code> - this is similar to a merge operation—two artifact versions can be moved into the same directory, but individual files with the same name will be overwritten.

Field Name	Description
Retrieved Artifact Location Property	<p>Name or property sheet path used by the step to create a property sheet. This property sheet stores information about the retrieved artifact versions as XML in the <code>artifactVersionXML</code> child property, and the file system location of the retrieved artifact version in the <code>cacheLocation</code> child property.</p> <p>The initial value of this field includes <code>[\$assignedResourceName]</code> as one of the levels in the property path because different resources may have different cache directories, and you do not want a retrieval on one resource to clobber the data regarding a retrieval on a different resource.</p> <p>Typically, a step that needs to use files from an artifact retrieved by this step will specify it in a Perl script similar to this:</p> <pre>my \$cmdr = new ElectricCommander(); my \$xpath = \$cmdr->getProperty ("/myJob/retrieveArtifactVersions/[\$assignedResourceName] /MyGrp:MyKey/cacheLocation"); my \$retrieveDir = \$xpath->findvalue("//value")->value (); system("java", "-cp", "\$retrieveDir/lib/bar.jar", "Bar");</pre> <p>This example shows how to retrieve the location of the retrieved <code>MyGrp:MyKey</code> artifact version on the resource, and add a file (<code>bar.jar</code>) in its <code>lib</code> directory to classpath, so the <code>Bar</code> class can be run.</p>
Filters	Enter search filters, one per line, applicable to querying the ElectricFlow database for the artifact version to retrieve.

Artifact Retrieval Dependency Order Explained

The order of dependencies registered for an artifact version are significant.

Consider this scenario:

- A depends on B (any version) and C [1.0, 2.0]
- B depends on C (any version)
- C versions 1.0, 2.0, and 3.0 exist

When retrieving A, the dependency algorithm evaluates B first. The algorithm finds that the max version of B depends on any version of C, so the algorithm looks for max version C and finds C 3.0. Because this chain is satisfied, the algorithm returns to A and evaluates its next dependency "C [1.0, 2.0)". This results in matching C 1.0.

The returned artifacts are: A, B, C 1.0, and C 3.0.

Consider if the A dependency is changed to:

- A depends on C [1.0, 2.0), B (any version)

The algorithm will choose C 1.0 first. Then the algorithm evaluates B, determines that its "C (any version)" is satisfiable by the already chosen C 1.0.

The returned artifacts are A, B, and C 1.0.

Note: In the version range syntax `[]` indicates inclusive, `()` indicates exclusive.

Send Email Step

Use the EC-SendEmail plugin to create a step to send email notifications.

The following parameters are available to create this step:

Parameter Name	Description
Email Configuration	The name of the email configuration to use. If no configuration is specified, the configuration named "default" is used.
To	List the "To" recipients for the email message, one per line. A recipient can be a user or group name or a complete email address.
CC	List the "CC" recipients for the email message, one per line. A recipient can be a user or group name or a complete email address.
BCC	List the "BCC" recipients for the email message, one per line. A recipient can be a user or group name or a complete email address.
Subject	The subject line for the email message.
Message	The body of the email message or a workspace file containing the body. If both a plain text and an HTML message are provided, both values are sent as alternates in a multipart message. If a raw message is provided, both values are sent as alternates in a multipart message. If a raw message is provided, the value should be a properly formatted RFC822 message.
Advanced options	
Multipart Mode	The multipart mode of the email message. Defaults to "none" unless there are multiple parts, in which case it defaults to "mixedRelated".
Header(s)	One or more RFC822 email header lines (for example: "reply-to: user@host.com").

Parameter Name	Description
Attachment(s)	One or more files from the job's workspace to send as attachments. The filename extension is examined to determine the content-type.
In-line Attachment(s)	One or more inline attachments specified as a <code>contentId</code> and a workspace filename. The filename extension is examined to determine the content-type.

New Extract Preflight Sources Step

Choose one of the preflight plugins to create a step to retrieve an artifact version. There are numerous source code management (SCM) plugins. For example, Accurev, Bazaar, CVS, ClearCase, Git, Mercurial, Perforce, SVN, StarTeam, Team Foundation Server, and Vault.

Enter the following information:

Field Name	Description
Step Name	Enter a unique name for your preflight step. You can use any name of your choice.
SCM Configuration	Use the pull-down menu to select a source control configuration. If a configuration for your source control type does not appear in the drop-down menu, select the Administration > Source Control tabs to configure your source control system.
Destination Directory	This field appears after specifying your SCM from the pull-down menu. If this field does not specify "Required", you do not need to enter this information. The Destination Directory is a path relative to the job's workspace, where the source tree will be created.

Click **Submit** after entering your information.

After clicking **Submit**, the Edit Step page is displayed.

Use the Edit Step page to specify additional information for your preflight step. Also, you can attach a credential, create an email notifier, add custom properties, and so on to your step. For assistance using the Edit Step page, access the **Help** link in the upper right corner of the page.

For more information about preflight builds, see the [Preflight Builds](#) Help topic.

Parameter - create new or edit existing parameter

To create a new parameter

Enter information in the fields as follows:

Field Name	Description / Action
Name	Enter a unique name to specify the parameter when a procedure or workflow is invoked.
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code>
Type - Select the parameter type from the drop-down menu. The following "types" are available.	
Text entry	Allows a short text entry in the Default Value field.
Text area	Expands the Default Value field to allow adding a longer script-type entry.
Dropdown menu	<p>Creates a drop-down menu from which to select a value when the parameter is presented.</p> <ul style="list-style-type: none"> Enter options - Click + Add Option to add a new row. Type-in the text and value for each option. The text is what will be displayed in the menu, and the value is the parameter value if that option is selected. Load options from list - Enter a pipe-separated list of options (for example, <code>foo bar baz</code>). The text and value for the options will be the same. Load options from property sheet - Enter the path to property sheet that contains options for the parameter. The property sheet must be created in a specific format: <ul style="list-style-type: none"> An <code>optionCount</code> property must exist whose value is the number of options. For each option, create a nested property sheet called <code>optionN</code>, where N is the option number, starting with 1. In each nested sheet, create two properties - <code>text</code> and <code>value</code>. The value of the property <code>text</code> will be displayed in the menu. The value of the property <code>value</code> is the parameter value if that option is selected. If <code>optionCount</code> is set to 3, you must create three nested sheets - <code>option1</code>, <code>option2</code>, and <code>option3</code>.

Field Name	Description / Action
Radio selector	<p>Creates "radio" buttons to select an entry when the parameter is presented.</p> <ul style="list-style-type: none"> • Enter options - Click + Add Option to add a new row. Type-in the text and value for each option. The text is what will be displayed in the menu, and the value is the parameter value if that option is selected. • Load options from list - Enter a pipe-separated list of options (for example, <code>foo bar baz</code>). The text and value for the options will be the same. • Load options from property sheet - Enter the path to property sheet that contains options for the parameter. The property sheet must be created in a specific format: <ul style="list-style-type: none"> • An <code>optionCount</code> property must exist whose value is the number of options. • For each option, create a nested property sheet called <code>optionN</code>, where N is the option number, starting with 1. • In each nested sheet, create two properties - <code>text</code> and <code>value</code>. The value of the property <code>text</code> will be displayed in the menu. The value of the property <code>value</code> is the parameter value if that option is selected. • If <code>optionCount</code> is set to 3, you must create three nested sheets - <code>option1</code>, <code>option2</code>, and <code>option3</code>.
Checkbox	<p>Creates a checkbox for a value to select (or not) when the parameter is presented.</p> <ul style="list-style-type: none"> • Value when unchecked - The value of the parameter when the checkbox is unchecked. • Value when checked - The value of the checkbox when the checkbox is selected. • Initially checked - Whether or not the checkbox should be checked initially. If true, set the "Default value" to match the "Value when checked". If false, set the "Default value" to match the "Value when unchecked".
Credential	<p>Selecting this option requires the user to specify a user name and password to use this parameter at run time.</p> <p>Note: Credential parameters are not supported on state definitions.</p>
Project	<p>Creates a project selector field on the Run Procedure page to choose a different project name where the parameter needs to find additional information.</p>

Field Name	Description / Action
Saved Filter	Use this option to reference any saved filters, which is a passing the filter into the procedure at run time (primarily used for reports). Saved Filters are stored as a property in your chosen project.
Default Value	You can specify a default value to assign to the parameter if no explicit value is provided.
Required?	Click the checkbox to select the parameter as "required." If the parameter is required, the procedure or workflow will not run without entering a value for that parameter. In this case, the default value is ignored. After the procedure or workflow begins execution, the parameter value is available in the job or workflow's property sheet with the same name as the parameter.
Defer Expansion?	Click the checkbox to set the parameter expansion to be "deferred." Deferred means the parameter value will not be expanded when the procedure call is expanded, but can be expanded from a command step instead.

Click **OK** after filling-in the fields.

To edit an existing parameter

You may change any information or add new information as necessary.

For example: Want to rename a parameter? Enter a new parameter name in the Name field, click **OK**.

For more information on property sheets, see the [Properties](#) Help topic.

Projects

This page displays all projects available on this ElectricFlowserver— the projects you create, ElectricFlow-supplied default projects, and ElectricFlow solutions. From this page you can navigate into projects to view its procedures, steps, schedules, credentials, workflows, and other components.

A project is a top-level container within ElectricFlow. Most information about software production processes, such as procedures, schedules, jobs, and workflows are contained within a project.

- The drop-down menu (at the top of the table) displays all "tags" you previously defined to mark, label, or group related or similar type projects. When you select a tag, the project list changes to display only the projects with that tag.

To add a tag:

- Click the **Edit** link for the project you want to tag.
- On the Edit Project page, enter a tag name in the Tags field.
- After creating a tag, the tag name will appear in the drop-down menu.

Note: For more information on "tagging" projects, see the [Edit Project](#) Help topic.

- To create a new project - click the **Create Project** link.
- **Edit, Copy, or Delete** an existing project - click a corresponding link in the Actions column.

Note: If deleting a project, the "background deleter" will mark the project for deletion, but the deletion process may not start until the server completes other tasks already queued or in progress. Deleting a project could take considerable time depending on the size of your project. For example, a project containing hundreds of jobs, steps, and other objects will take longer to delete than a simple project with 3 procedures and 10 steps.

- To see project details - click on a project name in the Project column to go to the Project Details page.

Projects have two purposes

First, projects allow you to create separate work areas for different purposes or groups of people so they do not interfere with each other. For example, different projects can reuse the same names internally without conflict, and each project has its own access control that determines who can use and modify the project. In a small organization, you might choose to keep all work within a single project, but in a large organization, you may want to use projects to organize information and simplify management.

Second, projects simplify sharing. You can create library projects containing shared procedures and invoke these procedures from other projects. After creating a library project, you can easily copy it to other ElectricFlow servers to create uniform processes across your organization.

ElectricFlow includes default projects

Currently, ElectricFlow includes the following 4 projects:

- EC-Utilities - this project contains procedures to perform basic ElectricFlow tasks. Also, these procedures can be used as examples - copy a procedure to another project and then modify it for your purpose. **Note:** By default, only the "admin" user has execute privileges on the EC-Utilities project. The admin user can enable privileges for additional users or groups.
- EC-Examples - this project contains templates for procedures that perform basic ElectricFlow tasks.
- Electric Cloud - this project contains procedures to manage ElectricSentry, specifically the Sentry Monitor for schedules and also contains ElectricFlow global reports.

Important notes:

When you create project names, do not use "Electric Cloud" or any other ElectricFlow-supplied project name. Duplicating a project name creates a conflict. Also, the "EC-" prefix is reserved for ElectricFlow-supplied project names.

Procedures within each project [above] are maintained by Electric Cloud.

ElectricFlow Solutions

ElectricFlow solutions are developed in the field by Electric Cloud SEs—our field team has the best experience in applying ElectricFlow "best practices" to solve real problems. In addition, ElectricFlow solutions can be contributed by ElectricFlow users from any customer site.

Potentially, any available solution can be considered a new ElectricFlow feature "waiting" to be included in the product if that solution gains wide, general appeal within the ElectricFlow customer base. However, because solutions are not part of the ElectricFlow product, and not developed by the ElectricFlow Engineering team, they are not supported by the Electric Cloud Technical Support team. If and when a solution becomes an integral part of the product, that solution will then have full Technical and Engineering support like any other ElectricFlow feature.

Currently, ElectricFlow solutions are posted to the Electric Cloud Forum Site at <http://forums.electric-cloud.com/ecforums/> - each solution, accompanied by its own installation instructions and Help file, can be downloaded from the forum site. If you have a solution you would like to contribute, the Electric Cloud / ElectricFlow Forum is where you need to post it.

When you install a solution on your ElectricFlow server, it will be displayed on the Projects web page as a project.

Installing an ElectricFlow Solution

Add new or upgrade existing ElectricFlow solutions by running the built-in setup script:

1. Download the ElectricFlow <solution name>.jar file from the Electric Cloud Community Forums web site.
For this example, the "solution name" is `Test`.
2. Save your download to a temporary directory on the system that hosts your ElectricFlow server.
3. Unpack the jar file in the temporary directory—the file creates a directory named `Test`.
 - If Java JDK is installed, use the included jar utility to unpack the .jar file: `%jar xf Test.jar`
 - For Linux, use any unzip utility, for example: `unzip Test.jar`
 - For Windows, use Windows Explorer to access the jar contents if you rename the file to `Test.zip`
4. Run the included `setup.pl` script in the jar through the `ec-perl` interpreter (installed in the `bin` directory). You must have "admin" or equivalent privileges to run the setup script.

```
ectool login admin
%cd <test>
%ec-perl setup.pl
```

Project - Create New or Edit Existing Project

Creating a New Project

Enter information in the fields as follows:

Field Name	Description
Name	Enter a unique project name. You may want your project names to describe the work groups or teams that will be using them. For example, you may set project names based on the products they support.
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> . . . </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code>
Default Resource	<p>Enter a resource name, or select Browse to find one. If you have not set up resource names or locations yet, click the Resources tab and then the Resources tab to see the Resources page.</p> <p>Note: This field is optional. If you specify a resource in this field, it will be used as the default for all jobs that run under the project. This is a convenient way to use a single resource for an entire project. You can change the resource by changing the value. There are several other places where you can specify resources, such as procedures and individual job steps. If you specify a resource in another location, it will take precedence over the one specified in this field.</p>
Default Workspace	<p>Enter a workspace name, or select Browse to find one. If you have not set up workspace names or locations yet, click the Resources tab and then the Workspaces tab to see the Workspaces page.</p> <p>Note: This field is optional. If you specify a workspace in this field, it will be used as the default for all jobs that run under the project. This is a convenient way to use a single workspace for an entire project. You can change the workspace by changing the value. There are several other places where you can specify workspaces, such as procedures and individual job steps. If you specify a workspace in another location, it will take precedence over the one specified in this field.</p>

Click **OK** to save the information you entered.

Your new project name will appear on the Projects page.

Editing an Existing Project

- To rename the project, enter a new name in the project Name field and click **OK**.
- You may modify the description or enter a text description if this field is blank.
- You can modify any previously entered information.

- If you would like to use a particular credential (by default) for all jobs that run under this project, select it here. If you leave this field blank, credential information from other sources is used.

Tags - (Optional) Enter a tag if you want to categorize or "mark" this project to identify its relationship to one or more other projects or groups. For example, you may want to tag a group of projects as "production" or "workflow", or you may want to use your name so you can quickly sort the project list to see only those projects that are useful to you.

- The tag name you enter in this field is displayed in the drop-down menu on the Projects page. When you select a specific tag from the drop-down menu, you see only the projects with that tag.
- Assigning a tag creates an "ec_tag" property that can be seen (with its value - the tag name you chose) after selecting the Properties subtab from the Project Details page,
- Do not use "spaces" in tag names. For example, if you want to mark a project with the tag name, "Eng Team A", you need to specify, "eng_team_a".
- Do not use symbols in a tag name—results are unpredictable.
- If a project is related to several project categories, you can enter a space-delineated list to add numerous tags at the same time.
Type a new name in the project Name field and click **OK**.
- To delete a tag, select the text and use your Delete key.

Impersonation Credential

Credential Project - The default Credential Project is the current project, or select the other button and Browse for the project name you need.

Credential Name - This is the name of the credential for the project. You can select **Browse** to find it.

Click **OK** after entering new or updated information.

For more information, see the [Credentials and User Impersonation](#) Help topic.

Project Details

This page displays project components, including procedures, workflow specifications, jobs, schedules, any credentials, properties, and reports.

Links and actions at the top of the page

- **Edit** - use this link to go to the Edit Project page.
- **Access Control** - use this link to go the Access Control page for this project.
- The "star" icon allows you to save this job information to your Home page.
- The "curved arrow" icon returns you to the Projects page.
- **Pagination** - Use the "previous" and "next" arrow icons to view the previous or next project. The numbers between the arrow icons display the number of projects you can view and the first number indicates which project [in the list] you are viewing.

The "tabbed" sections

The tabs at the top of the table allow you to select the type of information you want to see. See the 8 tabs illustrated in the following screen example:

Project Details / Electric Cloud				
Procedures Workflow Definitions Jobs Workflows Schedules Credentials Properties Reports				
Procedure +	Description	Resource	Create Date	Actions
runReports	Runs batch reports.	local	2015-07-12 06:11:01 PDT	

The Project Details page opens with the Procedures tab highlighted, so the first table you see is the Procedures table.

Procedures tab

The following links and actions are available in the Procedures table:

- Click the **Create Procedure** link (at the top of the table) to go to the New Procedure page to create another procedure.
- Click on a procedure name (first column) to go to the Procedure Details page for that procedure.
- Action column - click on any of the available links (**Run, Edit, Copy, or Delete**) to perform that function for the procedure listed in that row.

Note: The Run link provides two choices when you hover the mouse over the down-arrow. Select **Run Immediately** or **Run...**

- **Run Immediately** - This option runs the procedure "as-is" immediately.
- **Run...** - This option produces the Run Procedure web page where you can modify any existing parameters for this procedure, or set an existing credential.

Workflow Definitions tab

This tab provides a table listing all defined workflow definitions and includes the following functionality:

- Click the **Create Workflow Definition** link to go to the New Workflow Definition page to define additional definitions.
- Name column - Click on a workflow definition name to go to the Workflow Definition Details page (for that workflow definition).
- Action column
 - **Run** - use the down-arrow to select **Run...** or **Run Immediately**
 Selecting **Run...** takes you to the Run Workflow page where you can set the starting state for the workflow.
 Selecting **Run Immediately** runs the workflow "as-is", immediately.
 - **Copy** - use this link to make an exact copy of an existing workflow definition.
 - **Edit** - use this link to edit an existing workflow definition or just to change the name of the copy you created.
 - **Delete** - deletes an existing workflow definition on the same row.

Jobs tab

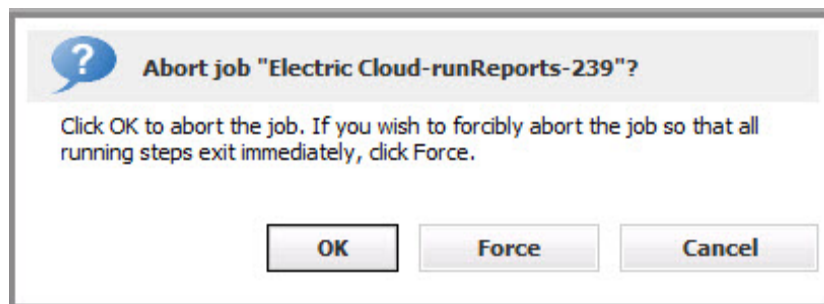
This tab provides a project-specific job table. Selecting the Jobs tab on the Project Details page displays only those jobs for the project you selected.

This table is a subset of the main Jobs tab that displays **all** jobs, regardless of the project of origin, and all Jobs page functionality is basically the same on either Jobs tab.

The following links and actions are available on this Jobs page:

- Job - click on Job to sort the column alphabetically or click on a job name to go to that job's Job Details page.
- Status - click on Status to sort this column by Running, Success, Warning, Error, or Aborted.
- Priority - displays the job's priority set by a "run procedure" command or by the job's schedule.
- Procedure - click on a Procedure name to go to the Procedure Details page for that procedure.
- Launched By - click on a name in this column to go to the page for the schedule (Edit Schedule page) or the user that ran the job.
- Elapsed Time - click Elapsed Time to sort the time values from longest to shortest time, or the reverse.
- Start Time - click Start Time to sort this column from the start time of the first job to the start time of the most recent job, or the reverse.
- Actions - use this column to Abort a running job or Delete a completed job. If your job has finished running, you will not see the Abort link.
 - Aborting a job requires the execute privilege on the job—not just the modify privilege.

When you click **Abort**, the following dialog box appears:



Select either **OK** or **Force** to abort the job, or **Cancel** if you change your mind.

Note: Deleting a job on this page causes removal of job information from the ElectricFlow database, but information in the job's on-disk workspace area is not affected. You must delete workspace information manually.

Workflows tab

The Workflows tab provides the following functionality:

- Name column - Click on a workflow name to go to its Workflow Details page.
- State column - Click on a state name to go to its State Details page.
- Modify Time - This time value represents the most recent workflow activity.
- Workflow Definition column - Click on a workflow definition name to go to the Workflow Definition Details page.
- Completed column - a "checkmark" in this column identifies this workflow as completed.
- Actions column - Click the Delete link to remove the workflow in that row.

For more information about workflows, see the [Workflow](#) Help topic.

Schedules tab

The Schedules tab provides the following functionality:

- To create a new schedule for this project, click either the Schedule link to create a new standard schedule to run at specific and/or days or the CI Configuration link to setup a new continuous integration schedule.
 - Choosing "Schedule" displays the New Schedule page.
 - Choosing "CI Configuration" displays the New CI Configuration page.

Table column descriptions

- To edit an existing schedule, click on the schedule name (first column).
- Select the check box in the Enabled column to enable or disable the schedule in that row.
- The Priority column displays the job's priority you selected while creating the schedule for this procedure.
- Action column links -
 - **Run** - runs the schedule with the permissions of the logged in user.
For example, if this schedule is timed to run every day at 11:00 pm, you might decide to run the schedule at 6:00 pm on a particular Thursday, without changing the regular schedule settings. However, if you do not have the appropriate permissions to run this schedule, it will fail.
 - **Copy** - makes a copy of the schedule on that row.
 - **Delete** - deletes the schedule in that row.

The following screen example illustrates the Schedules table:

Project Details / Electric Cloud									
<div> Procedures Workflow Definitions Jobs Workflows Schedules Credentials Properties Reports </div> <div>Create: Schedule CI Configuration</div>									
Schedule	Enabled	Procedure	Priority	Description	When to run	Time Zone	Actions		
ECSCM-SentryMonitor	<input type="checkbox"/>	ECSCM-ElectricSentry	normal	Periodically locate ECSCM sentry schedules to run	Run every 5 minutes from 07:00 to 23:00.	America/Los_Angeles			
Report Recent Job Outcome	<input type="checkbox"/>	EC-Reports:RunReport_CategoryPie	normal	Pie chart displaying the distribution of all job outcomes in the past 14 days	Run at 23:45.	America/Los_Angeles			
Report Recent Job Trend	<input type="checkbox"/>	EC-Reports:RunReport_MultiSeries	normal	Multi-line chart displaying the trend of all job outcomes in the past 14 days	Run at 23:50.	America/Los_Angeles			
ReportSchedule	<input type="checkbox"/>	runReports	normal	Run data collection, summarization, and report generation	Run at 02:00.	America/Los_Angeles			

For information on creating a standard schedule, see the [Schedule - create new or edit existing schedule](#) Help topic.

For information about CI Manager and adding a configuration, click the Home tab, the Continuous Integration subtab, then click the Help link in the upper-right corner of the page.

Credentials tab

The Credentials tab provides the following functionality:

- To create a new credential for this project, click the **Create Credential** link at the top of the table.
- To edit an existing credential, click on the credential name (first column).
- Action column - the Delete link deletes the credential on that row.

For information about credentials and access control, see the [Credentials and User Impersonation](#) or [Access Control](#) Help topics.

Properties tab

The Properties tab provides the following functionality:

- To create a new property for this project, click the **Create Property** or **Create Nested Sheet** link.
- To view or change privileges on the property sheet, click the **Access Control** link.
- To edit an existing property, click on the property name (first column).
- Action column - the Delete link deletes the property on that row.

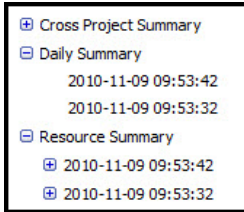
For more information about properties, see the [Properties](#) Help topic.

Reports tab

Selecting this subtab yields different results, depending on whether or not you have installed the additional ElectricFlow-supplied optional reports or have created custom reports for the specific information you need. Generally, if you have not yet configured additional reports, you will see the three installed, default ElectricFlow reports (Cross Project Summary, Daily Summary, and Resource Summary) as illustrated in the screen example below.

- When a report runs, the report "folder" is populated with the completed report. Reports are listed by date and time.
- Each individual report "date/time" entry is a link to see the full report.
- To create a new report, click the **Create Report** link to go to the web page to create a Multiple Series, Category, Count Over Time, or Procedure Usage report.

This is an example of the information on the Reports page :



For more information about reports, see the [Reports](#) Help topic series.

Tips

Want to rename this project?

Click the **Edit** link at the top of this page and enter a new name.

Want to rename a procedure in this project?

Click the **Edit** link for the procedure (on this page) and enter a new name.

Run Procedure

Using the Run Procedure page

- You may see a message informing you that no parameters were defined for this procedure. If this is the case, click **Run** to run the procedure.
- If parameters were defined for this procedure, you will see specific fields based on those parameters. You may make changes to any field you choose if you want to run this procedure under different parameters at this time. However, any changes you make are specific to this run procedure only. Your changes on this web page will not affect previously defined parameters for this procedure. Click **Run** to run the procedure.
- You may see a "Required" field. To run the procedure, you must enter the required parameter previously defined on the Procedure Details page.

Click **Run** to run the procedure.

Advanced section

Priority:

Use the drop-down menu to select the priority for this run procedure. Available priorities are: low, normal (default), high, or highest.

- You can select the job's priority here on the Run Procedure web page or when you schedule the procedure to run at a regular time or interval. When a job is launched, its priority cannot be changed.
- Priorities take effect when two or more job steps in different jobs are waiting for the same resource. When the resource is available, it will be used by the job step that belongs to the job with the highest priority. If the priority level is the same, the resource will be used by the job step that belongs to the job with the lowest job ID number. If the job steps are in the same job, the resource will be used first by the step with the lowest job step ID number.
- If using ectool, the priority can be set by passing "--priority=<low|normal|high|highest>" to a `runProcedure`, `createSchedule`, or `modifySchedule` operation.

Note: To raise the job priority level above "normal," the user who launches the procedure must have execute permission on the system-level access control list for priorities.

Impersonation:

Select one of the following:

- **Use pre-defined credential** - (Default option) Use this option if you have not defined a credential for this procedure.
ElectricFlow looks for a credential first in the top-level procedure and then in its project. If it finds a credential, it uses the login from that credential as the default for the job. If there is no credential in either place, by default, the job's steps run as the user under which the agent for the step is running.
- **Use specific credential** - If you select this option, more fields are displayed for you to choose a particular credential and enter its name from those defined for the project containing the procedure. The login user from that credential will be used as the default for the job.
- **Use a specific user** - If you select this option, more fields are displayed to enter a User Name and Password. The specified login is used as the default for the job.

Click **Run** to run the procedure.

For more information on credentials and user impersonation, see the [Credentials and User Impersonation](#) Help topic.

Schedule - create new or edit existing schedule


Use this page to create or edit a standard schedule that specifies a set of times and days when a particular procedure should run. If you need to create a continuous build integration instead, see the Help information available from the Continuous Integration Dashboard web page.

For any schedule, click the "star" icon to add this schedule to the Shortcut section on your Home page.

To create a new standard schedule

Enter information in the fields as follows:

Field Name	Description
<i>General section</i>	
Name	Enter a Schedule name. This must be a unique name among other schedule names in this project.
Procedure	Displays the current project and procedure. Click Change to use the pop-up menu to select a new project or procedure.

Field Name	Description
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code>
<p>Parameters section</p> <p>If the procedure has parameters, enter information in their values in this section. The procedure cannot execute unless all required parameters are provided. If you move the mouse over a field for a particular parameter, inline Help displays the description of that parameter if one was provided in the procedure's definition.</p>	
	Click this icon to displays fields to add one or more parameters.
<p>Frequency section</p>	
Run at ???	This is a summary section: As you make selections for the following Days and Repetition sections, a summary of what you have selected appears here.
Days	<p>Select weekdays, month days, or create a custom frequency when you want the schedule to run.</p> <p>Note: The "day" applies to the start time if the time range spans two days, that is, if the time range crosses the midnight boundary.</p>
Repetition	<p>If Run Once - the schedule will run once per selected day, at the time you specify.</p> <p>If Run Every - the procedure will execute repeatedly during the specified time range. For example, if you select a time range from 11:00-14:00 and an interval of 40 minutes, the procedure will execute 5 times on each of the selected days: at 11:00am, 11:40am, 12:20pm, 1:00pm, and 1:40pm.</p> <p>If Run Continuously - as soon as one job ends, the scheduler will trigger immediately to run the job again.</p>
<p>Advanced section</p>	
Enabled	If this box is "checked," the schedule will run. You can disable this schedule whenever necessary.

Field Name	Description
Misfire Policy	<p>Choose between two options to manage how a schedule resumes in cases where the normal scheduled time is interrupted. The server may not be able to trigger a job at the scheduled time for several reasons:</p> <ol style="list-style-type: none"> 1) a running job extends past the next scheduled run time, or 2) the server cannot find enough available resources and a delay occurs, or 3) the server is down or disconnected from the network for a period that overlaps one or more scheduled times. If any of these situations occur, the misfire policy defines two possible responses: <p><code>skip</code> - all misfires are ignored and the job runs at the next scheduled time.</p> <p><code>run once</code> - after one or more misfires, the job runs at the soonest time that occurs within an active region. For example, at server startup a scheduled job will not trigger immediately on startup.</p>
Time Zone	<p>Default is the time zone of the ElectricFlow server, or select a county from the first pull-down menu, then select an area from the second pull-down menu. Another option: Select "other" from the first pull-down menu, then select GMT [or another selection] from the second pull-down list.</p>
Priority	<p>Use the pull-down menu to select one of four priorities for this procedure.</p> <p>Also, you can select the job's priority on the Run Procedure web page. When a job is launched, its priority cannot be changed.</p> <p>Priorities take effect when two or more job steps in different jobs are waiting for the same resource. When the resource is available, it will be used by the job step that belongs to the job with the highest priority.</p> <p>If the priority level is the same, the resource will be used by the job step that belongs to the job with the lowest job ID number. If the job steps are in the same job, the resource will be used first by the step with the lowest job step ID number.</p> <p>If using ectool, the priority can be set by passing</p> <p><code>--priority=<low normal high highest></code> to a <code>runProcedure</code>, <code>createSchedule</code>, or <code>modifySchedule</code> operation.</p> <p>Note: Schedules are launched by the project principal user. To raise the priority above "normal," this user must have execute permission on the system-level access control list for priorities.</p>

Field Name	Description
<p><i>Impersonation Credential section</i></p> <p>If you select a credential for the schedule, it will be used as the default for jobs run from the schedule. For example, job steps will run under the login account specified in the credential. However, the schedule's credential is low in priority: it will be used only if there is no credential specified in the procedure or its project. Also, individual steps can override the credential with a specific credential for that step.</p> <p>For more information, see the Credentials and User Impersonation Help topic.</p> <p>Impersonation credentials are used to set the top-level impersonation credential for a job. If specified, the impersonation credential [on the job] is used as the default impersonation credential for all steps in the job.</p>	
Credential Project	The name of the credential for the project. By default, the current project is selected. Enter a different project name or select Browse to find this name.
Credential Name	The name of the credential for the project - you may need to Browse to find this name.

Click **OK** to save your schedule information.

To create a new continuous integration "schedule"

If you selected the **CI Configuration** link, the New CI Configuration page is displayed. **For information** on populating these fields, see the "Continuous Integration Manager" Help topic - click the **Help** link in the upper-right corner of the New CI Configuration page.

See the Standard Schedule information [above] to enter information in the fields common to both types of schedules.

To edit an existing schedule

Modify any of the information you previously specified.

The following additional links are available at the top of the page:

- **Save Configuration** - if you click this link, the information in this schedule will be copied into a new Job Configuration and displayed on your Home page for easy access.
- **Access Control** - click this link if you need to set permissions for this schedule.

Attached Credentials

Click the **Attached Credential** link if you need to attach an existing credential to this schedule. If you need to create a new credential for this schedule, return to the Project Details page.

Custom Schedule Properties

Select the **Create Properties**, **Create Nested Sheet**, or **Access Control** links if you want to assign properties to this schedule.

For more information on Properties or Access Control, see the following main Help topics: [Properties](#), [Access Control](#)

Click **OK** to save your modifications for this schedule.

Credentials - create new or modify existing credential

To create a new credential

Enter information in the fields as follows:

Field Name	Description
Name	Enter a unique name of the credential object.
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> . . . </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code>
User Name	Enter the user name you choose to use for login.
Password	Enter the password that identifies the user account.

Click **OK** to save your new credential information.

To edit a credential

Modify any previously entered credential information and click **OK** to save your changes.

Use the **Access Control** link to add or change existing permissions.

You can add Custom Credential Properties for this credential also. Click the **Create Property**, **Create Nested Sheet**, or **Access Control** link in this section if you need to specify any of these items.

For more information, see the following Help topics:

[Credentials and User Impersonation](#)

[Properties](#)

[Access Control](#)

Properties

[Creating or modifying properties](#)

[Using property values](#)

[Property sheets and intrinsic properties](#)

[Property names and paths](#)

[Context-relative "shortcuts" to property paths](#)

[Property path shortcuts in ElectricFlow 5.0 and later on page 736](#)

[Property name substitutions](#)

[Expandable properties](#)

[The property hierarchy](#)

[Special property references](#)

[Intrinsic properties listed by object type](#)

[ElectricFlow Object / Property tables](#)

Overview

ElectricFlow provides a powerful data model based around the notion of a *property*.

- A property is a string value with a name.
- Properties can have arbitrary names and values.
- You can attach properties to any object in the ElectricFlow system, such as a project, procedure, or job.
- After a property is created, it is stored in the ElectricFlow database.
- You can retrieve and modify the property value later.
- You can delete properties that you no longer need.

Properties are used extensively throughout the ElectricFlow system and provide a flexible and powerful mechanism to manage data about your builds.

Property Use Case Examples

- When a job starts, it computes a unique identifier for that build and saves it in a property. Later build steps can retrieve the property value to embed the build number in binaries generated during the job.
- When a build executes, it can set a property on the procedure to identify the source code version used for the build (for example, a tag or timestamp from your source code control system). The next time the build executes, it can retrieve the property value from the procedure and use it to extract information from your source code control system about the files that changed since the last run.

- Suppose you have a collection of machines for testing, and some machines have older test hardware and some machines have newer hardware with slightly different characteristics. You can set a property on each test machine resource to indicate which version of test hardware is available on which machine. Later, when a test executes, the test can retrieve the property for the machine where it ran and configure tests appropriately for that hardware. If you upgrade test hardware on a machine, all you need to do is change the property on that resource to reflect the new version.
- You can set job properties to indicate the build status produced by that job. For example, if your QA team finds fatal flaws in a build, it can mark the job accordingly. Builds that need to be preserved (release candidates or builds undergoing beta testing at customer sites) can be marked with properties so those builds are not deleted.
- When a job executes, properties are set for its job steps that hold metrics such as how many files compiled during a build step, how many tests executed during a test step, or how many errors were detected in the step. These property values are included in reports and can be examined later to compute trends over time. You can define additional properties for metrics useful to you.

ElectricFlow provides Intrinsic properties and allows you to create Custom properties. This Help topic enumerates properties available within ElectricFlow (Intrinsic properties), distinguishes between relative and absolute property paths, and describes property hierarchies.

Note: Intrinsic properties are case-sensitive. Custom properties, like all other object names in the ElectricFlow system, are "case-preserving," but *not case-sensitive*.

- **Intrinsic properties**

These properties represent attributes that describe the object to which they are attached, and are provided automatically by ElectricFlow for each similar type object. For example, every project has a `Description` property that can be referenced with a non-local property path such as `/projects/Examples/description`.

- **Custom properties**

Custom properties are identical to intrinsic properties and when placed on the same object, are referenced in the same manner, and behave in every way like an intrinsic object-level property with one exception: they are *not* created automatically when the object is created. Instead, custom properties can be added to objects already in the database before a job is started, or created dynamically by procedure steps during step execution.

Creating or modifying properties

You can set a property value by using one of the following three methods or applications: the automation platform web interface, the **ectool** command-line application, or the Perl API.

- Using the web interface, you will see a table labeled Custom Properties on the pages that display details for projects, procedures, and so on.
Click **Create New Property** to create a new property for that object, or click on an existing property to change its value.
- Using the ectool application, set a property value with a command like:

```
ectool setProperty /myJob/installStatus complete
```

This command sets the property named `/myJob/installStatus` to the value `complete`, and creates the property if it did not already exist (the meaning of property names like `/myJob/installStatus` is

explained below). You can use `ectool` from within one job step to set properties accessed by later steps in the same job.

- Using the Perl API, you can use an external script or a script in a step with `ec-perl` as the shell. A Perl code example:

```
use ElectricCommander ();
my $ec = new ElectricCommander;
$ec->setProperty ("/myJob/installStatus", "complete", {jobStepId =>
    $ENV{COMMANDER_JOBSTEPID})
```

Note: Using the Perl API may yield better performance if you are requesting multiple API calls in one step.

For more information, see the "Using the ElectricFlow API" topic in the ElectricFlow API Guide.

Using property values

A job step can access a property value by two methods. The first method is *substitution*, using the `$()` notation. Suppose you enter the following text as a command for a step:

```
make PLATFORM=$(platform)
```

Before running the step, ElectricFlow finds the property named `platform` and substitutes its value in the command string in place of `$(platform)`. For example, if the property contains the value "windows", the actual command executed is:

```
make PLATFORM=windows
```

The substitution method can be used for a command in a step and for any step fields, such as the resource or working directory. This method allows you to compute configuration information in an early step of a job, then use that configuration information to control later steps in the job.

Another way for a step to access the property value is with the `ectool getProperty` command. For example, the following command returns the property value named `platform`:

```
ectool getProperty platform
```

Property sheets and intrinsic properties

A property value can be a simple string or a nested *property sheet* containing its own properties. Property sheets can be nested to any depth, which allows you to create hierarchical collections of information.

Most objects have an associated "property sheet" that contains "custom properties" created by user scripts. The property sheet is an intrinsic property of the containing object called "property sheet", so to reference a project's custom property "branchName", you could specify `"/projects/aProject/propertySheet/branchName"`.

As a convenience, ElectricFlow allows the "property sheet" path element to be omitted and the path written as `"/projects/aProject/branchName"`. If there is no intrinsic property with the same name, the path will find the property on the property sheet.

Custom properties in a property sheet can be one of two types: *string property* or a *property sheet* property. String properties hold simple text values. Property sheet properties hold *nested* properties. Nested properties are accessed via the property sheet property of their containing sheet.

For example:

```
/projects/aProject/propertySheet/topSheet/propertySheet/propB
- or -
/projects/aProject/topSheet/propB
```

All information managed by ElectricFlow exists in the form of properties and property sheets and your own custom-created properties. For example, each project, procedure, and step is represented internally as a property sheet—the command for a step is actually a property associated with the step, and so on. Every value in the ElectricFlow system can be accessed as a property, using the naming facilities described below. These properties are called *intrinsic* properties. *ElectricFlow enforces some restrictions on intrinsic property values, whereas custom properties can have any value you choose.*

To learn more about intrinsic properties defined for an object by ElectricFlow, see the "Using the ElectricFlow Perl API" topic in the ElectricFlow API Guide. For example, to learn about intrinsic properties associated with each project, find the documentation for the `getProject` ectool command. The result of running this command is an XML document whose field names and values represent the properties for the project.

Note: At the end of this Help topic, find the current list of ElectricFlow intrinsic properties sorted by each object and its associated properties [in table format].

Property names and paths

Properties are named using multi-level paths such as `first/second/third`, which refers to a property named "third" in a property sheet named "second" in a property sheet named "first." ElectricFlow also supports an equivalent notation using brackets instead of slashes. In this format, slashes are not considered separators when they appear between brackets.

For example, `first[second]/third` and `first[second][third]` both refer to the same property as `first/second/third`. The bracket `[]` notation is based on matching brackets. For example, `steps[build[1]]` refers to a property named "build[1]" inside a property sheet named "steps" (it does not treat "build" as a property sheet containing a property named "1").

Property names/paths have two forms: *absolute* and *relative*.

Note: Some properties can be named by specifying the property name and a globally unique identifier for the property. For example, you can specify a job step using

```
/jobSteps/<jobStep ID>/
```

even though `jobSteps` is not a top-level (absolute) property name.

Absolute property paths

Absolute property paths are referenced by a fully-qualified path syntax that begins with a slash character ("/") followed by a top-level name. This path syntax is similar to a file system path specification. The first component after the "/" must be one of several reserved words that select a starting location to look up the property name. For example, consider the property name `/server/Electric Cloud/installDirectory`. `/server` means

"lookup" starts in the topmost property sheet associated with the ElectricFlow server. This property name refers to the "installDirectory" property inside the server property sheet.

The system defines the following top-level names (absolute property names):

`/applications/...`

Start in the property sheet containing all applications. For example, `/applications[deploy]` refers to an application named "deploy" and `/applications[deploy]/processes[install]` refers to a process named "install" in that application.

`/artifacts/...`

Start in the property sheet containing all artifacts. For example, `/artifacts/myGrp:myKey/prop1` refers to the `prop1` property on the artifact whose name is "myGrp:myKey".

`/artifactVersions/...`

Start in the property sheet containing all artifact versions. For example, `/artifactVersions/myGrp:myKey:1.0-36/prop1` refers to the `prop1` property on the artifact version whose name is "myGrp:myKey:1.0-36".

Note: Throughout the API, you can substitute "groupId:artifactKey:version" wherever an `artifactVersionName` argument can be specified. This is the same if the `artifactVersionNameTemplate` specified in the artifact is in the form "groupId:artifactKey:version". When the template is different, it is convenient to be able to specify this tuple if you do not know the `artifactVersionName`.

`/components/...`

Start in the property sheet containing all components. For example, `/components[warfile]` refers to a component named "warfile" and `/component[warfile]/processes[backup]` refers to a component process named "backup" in that component.

`/environments/...`

Start in the property sheet containing all components. For example, `/environments[qeserver]` refers to an environment named "qeserver".

`/groups/...`

Start in the property sheet containing all groups. For example, `/groups[dev]` refers to the group named "dev".

`/jobs/...`

Start in the property sheet containing all jobs. For example, `/jobs[ecloud.4096]` refers to the job named "ecloud.4096." You can name a job using either its name (as in the preceding example) or using the unique identifier assigned to it by ElectricFlow.

`/plugins/...`

Start in the property sheet containing all plugins. For example, `/plugins[EC-AgentManagement]` refers to the currently promoted plugin named "EC-AgentManagement" and `/plugins[EC-`

`AgentManagement]/project/procedures[scpCopyFile]` refers to a procedure named "scpCopyFile" in that plugin.

Note: There is a subtle difference between

```
ectool setProperty /plugins/EC-AgentManagement/project/foo 'bar'
```

and

```
ectool setProperty /plugins/EC-AgentManagement/foo 'bar'
```

The former places the property on the plugin's project and can be referenced by `$/myProject/foo` while the latter places the property on the plugin object and will not be found via `$/myPlugin/foo`.

`/projects/...`

Start in the property sheet containing all projects. For example, `/projects[nightly]` refers to the project named "nightly," and `/projects[nightly]/procedures[main]` refers to a procedure named "main" in that project.

`/repositories/...`

Start in the property sheet containing all artifact repositories. For example, `/repositories/repo1/prop1` refers to the `prop1` property on the repository whose name is "repo1".

`/resourcePools/...`

Start in the property sheet containing all resource pools. For example, `/resourcePools[linuxA]` refers to the resource pool named "linuxA".

`/resources/...`

Start in the property sheet containing all resources. For example, `/resources[linux1]` refers to the resource named "linux1".

`/server/...`

Start in the top-level server property sheet. For example, `/server/a` refers to the server property named "a".

`/users/...`

Start in the property sheet containing all users. For example, `/users[Bob]` refers to the user named "Bob".

`/workspaces/...`

Start in the property sheet containing all workspaces. For example, `/workspaces[default]` refers to the workspace named "default."

Relative property paths

Property names that do not begin with `"/` are *relative* and looked up starting in the *current context*. For example, when executing a job step, the current context includes properties defined for the job step, parameters for the current procedure, and global properties on the current job.

Relative property paths are distinguished from absolute property paths because they do not begin with one of the top-level names. To avoid having to construct full property paths, ElectricFlow supports the concept of a relative property path.

In use, the relative property path value is resolved by its context and a defined search order, which results in accessing the value of an absolute fully-specified property value. Contexts and search orders are as follows:

1. In a job step context, ElectricFlow searches for relative property paths in the following order:
 - a property on the job step object
 - a property of the parent job step object, which includes parameters of the procedure on which the step is defined
 - a property on the job object

Notes:

- The search can be enabled or disabled by using the `--extendedContextSearch` option on the `ectool getProperty` or `setProperty` commands. When searching for a property value, disable the search by setting the `--extendedContextSearch` switch to "false", requiring the property to exist on the job step object or return false. When writing a new value to a property, enable the search by setting the `--extendedContextSearch` switch to "true", allowing the search for a property within the search order before creating a new one. New properties are created on the job step object.
 - Parameters for subprocedure calls from job steps are searched for in a job step context.
 - For procedure parameters for nested subprocedures, properties referenced by parameters are looked up as described [above] for job steps.
2. When expanding schedule parameters, ElectricFlow searches the relative property path in the following order:
 - a property on the procedure being called
 - a property on the project on which the procedure being called is defined
 - a property on the server
 3. In a job name template context, ElectricFlow searches for the relative property path as a property on the job.
 4. In any other context, ElectricFlow searches for the relative property path as a property on the context object.

Context-relative "shortcuts" to property paths

A shortcut can be used to reference a property without knowing the exact name of the object that contains the property. You might think of a shortcut as another part of the property hierarchy. Shortcuts resolve to the correct property path even though its path elements may have changed because a project or procedure was renamed. Shortcuts are particularly useful if you do not know your exact location in the property hierarchical tree.

Shortcuts to property paths that provide convenient runtime access include:

```
/myApplication/...
```

Start in the property sheet for the `application` associated with the current job or job step.

```
/myApplicationTier/...
```

Start in the property sheet for the `applicationTier` associated with the current job step.

`/myArtifactVersion/...`

Start in the property sheet for the `artifactVersion` associated with the current context. The only context where this property has any value is in the `artifactVersionNameTemplate` field for the artifact.

`/myComponent/...`

Start in the property sheet for the `component` associated with the current context.

`/myCredential/...`

Start in the property sheet for the `credential` associated with the current job step. This form produces an error if the current job step does not have a credential.

`/myEvent/...`

This is a special property used only within an "email notifier." `/myEvent/` allows you to refer to fields associated with the notifier itself. You can include these properties, for example, in the text of the email you send out for notification:

`/myEvent/notifier` - This property contains the name of the notifier that created the notifier event. You created this name when you created the notifier.

`/myEvent/entity` - This property contains the object where the notifier is attached. Two possible values are "job" or "jobStep," depending on where the notifier is attached.

`/myEvent/source` - This property contains the name of either the job or the jobStep where the notifier is attached.

`/myEvent/type` - This property defines whether the notifier is an "On Start" or an "On Completion" notifier. Two possible values are "STARTED" or "COMPLETED".

`/myEvent/time` - This property contains the time when the notifier occurred. The time is always specified in GMT and uses a formatted string, for example,
2009-06-11T21:00:56.502Z

`/myEvent/timeMillis` - This property contains the "timestamp" in milliseconds when the notifier occurred. This value is the number of milliseconds since January 1, 1970 GMT. The `timeMillis` corresponding to the time in the previous example is: 1244754056502

`/myEnvironment/...`

Start in the property sheet for the `environment` associated with the current job or job step.

`/myEnvironmentTier/...`

Start in the property sheet for the `environmentTier` associated with the current job step.

`/myJob/...`

Start in the global property sheet for the current job. `/myJob/` points directly to the job object so you can reference built-in job properties (`jobName`, `createTime`, `outcome`, and so on) and custom properties. Also, this property sheet can be used to hold working data that needs to be passed from one step to another within the job.

`/myJobStep/...`

Start in the property sheet for the current job step.

`/myParent/...`

Start in the global property sheet for the parent job step or job if this is a top-level step. `/myParent/` points directly to the job or job step object, so you can reference intrinsic or custom properties. For example, you can reference the parameters that were passed to this procedure.

In addition, you can reference a sibling step by calling `/myParent/jobSteps/<sibling step name>`.

Or, you can create additional properties on `/myParent/` to pass information from one step in the procedure to another:

step1 calls `setProperty /myParent/results 100`

step2 can call `getProperty /myParent/results`

`/myParent/`.

`/myProcedure/...`

Start in the property sheet for the procedure in which the current job step was defined. If the current job step is executing as part of a nested procedure, `/myProcedure/` refers to the innermost nested procedure.

`/myProcess/...`

Start in the property sheet for the application or component process in which the current job or job step was defined.

`/myProcessStep/...`

Start in the property sheet for the application or component process step in which the current job step was defined.

`/myProject/...`

Start in the property sheet for the project in which the current job step was defined. If the job step has nested procedure invocations, this is the project associated with the innermost nested procedure, for example, the project associated with `/myProcedure/`.

`/myResource/...`

Start in the property sheet for the resource assigned to the current job step.

`/myResourcePool/...`

Start in the property sheet for the resource pool that provided the resource for the current job step. Returns null if the step did not specify a resource pool.

`/myStep/...`

Start in the property sheet for the current step. "Step" refers to the (static) definition of a step, which is part of a procedure— this is different from a job step, which represents a step when it executes (dynamically) in a job. Use `/myJobStep/` to access the job step.

`/myState/...`

Start in the property sheet for the state object so you can reference built-in and custom state properties or find parameter values passed to that state. When accessed from a state, `/myState/` refers to that state. When accessed from a transition, `/myState` refers to the transition's owning state. When accessed from a job or job step, `/myState/` refers to the state that launched that job as a subjob.

`/mySubjob/...`

Start in the property sheet for the subjob so you can reference built-in and custom job properties, parameters passed to the job, and properties on steps within that job. When accessed from a transition, `/mySubjob/` refers to the subjob started by the state that owns that transition. This property path is particularly useful in conditions for On Completion transitions because the outcome or other information for the subjob can influence which state the workflow transitions to next.

`/mySubworkflow/...`

Start in the property sheet for the subworkflow so you can reference built-in and custom workflow properties. When accessed from a transition, `/mySubworkflow/` refers to the subworkflow started by the state that owns that transition. This property path is particularly useful in conditions for On Completion transitions because the active state and other information for the subworkflow can influence which state the workflow transitions to next. You can access information about states and transitions belonging to the workflow by using the path `/mySubworkflow/states/someState` or `/mySubworkflow/states/someState/transitions/someTransition`.

`/myTransition/...`

Start in the property sheet for the transition object so you can reference intrinsic and custom transition properties. `/myTransition/` is accessible only from a transition.

`/myUser/...`

This property can be used only if the current session is associated with: the predefined "admin" user, a user defined as "local", or a user defined by a Directory Provider (LDAP or ActiveDirectory). This property cannot be used if the user is a "project principal", which is normally the case when running inside an ElectricFlow step.

For example, with an interactive login you can use:

```
ectool getProperty /myUser/userName (to get the user name of the logged in user, or...)
ectool getProperty /myUser/email (to get the email address)
```

`/myWorkflow/...`

Start in the property sheet for the workflow object so you can reference built-in and custom workflow properties. When accessed from a state or transition, `/myWorkflow/` refers to the "owning" workflow. When accessed from a job or job step, `/myWorkflow/` refers to the workflow whose state launched that job as a subjob. You can access information about states and transitions belonging to the workflow by using the path `/myWorkflow/states/someState` or `/myWorkflow/states/someState/transitions/someTransition`.

`/myWorkflowDefinition/...`

Starts in the property sheet for the workflow definition object so you can reference built-in and custom workflow properties.

/myWorkspace/...

Start in the property sheet for the workspace associated with the current job step.

Property path shortcuts in ElectricFlow 5.0 and later

The following shortcuts to property paths are available in ElectricFlow 5.0 and later. For more information, go to [Context-relative "shortcuts" to property paths](#) on page 732.

Shortcuts	Descriptions	For jobs	For job steps
/myApplication/...	Starts in the property sheet for the application associated with the current job or job step.	Yes	Yes
/myApplicationTier/...	Starts in the property sheet for the application tier associated with the current job step.	No	Yes
/myComponent/...	Starts with the property sheet for the component associated with the current job step.	No	Yes
/myCredential/...	Starts with the property sheet for the credential associated with the current job step.	No	Yes
/myEnvironment/...	Starts in the property sheet for the environment associated with the current job or job step.	Yes	Yes
/myEnvironmentTier/...	Starts in the property sheet for the environment tier associated with the current job step.	No	Yes
/myJob/...	Starts in the property sheet for the job associated with the current job or job step.	Yes	Yes
/myJobStep/...	Starts in the property sheet for the job step associated with the current job step.	No	Yes
/myParent/...	Starts in the global property sheet for the parent job step.	No	Yes
/myPlugin/...	Starts in the property sheet for the plugin associated with the current job.	Yes	No
/myProcedure/...	Starts in the property sheet for the procedure in which the current job or job step was defined.	Yes	Yes

Shortcuts	Descriptions	For jobs	For job steps
<code>/myProcess/...</code>	Starts in the property sheet for the process in which the current job or job step was defined.	Yes	Yes
<code>/myProcessStep/...</code>	Starts in the property sheet for the process step in which the current job step was defined.	No	Yes
<code>/myProject/...</code>	Starts in the property sheet for the project in which the current job was defined.	Yes	No
<code>/myResource/...</code>	Starts in the property sheet for the resource associated with the current job step.	No	Yes
<code>/myResourcePool/...</code>	Starts in the property sheet for the resource pool associated with the current job step.	No	Yes
<code>/myState/...</code>	Starts in the property sheet for the state object so you can reference built-in and custom state properties or find parameter values passed to that state.	Yes	Yes
<code>/myStep/...</code>	Starts in the property sheet for the step associated with the current job step.	No	Yes
<code>/myWorkflow/...</code>	Starts in the property sheet for the workflow object so you can reference built-in and custom workflow properties.	Yes	Yes
<code>/myWorkflowDefinition/...</code>	Starts in the property sheet for the workflow definition object so you can reference built-in and custom workflow properties.	Yes	Yes
<code>/myWorkspace/...</code>	Starts in the property sheet for the workspace associated with the current job step.	No	Yes

Property name substitutions

Property names can contain references to other properties, which are then substituted into the property name before looking it up. For example, consider the following property name:

```
/myStep/${/myProcedure/name}
```

If the value of `/myProcedure/name` is "xyz", the property above is equivalent to `/myStep/xyz`.

Expandable properties

Property values can contain property references using the `"${}"` notation.

For example:

1. Create a property named "foo" with a value of `hello ${bar}`.
2. Create a property named "bar" with a value of `world`.
3. Reference "foo" (either using `${foo}` or `ectool getProperty foo`). The value "hello world" is returned.

If you want just the literal value of "foo" (useful in the UI, for example), you can use the `expand` option in `ectool`:

```
ectool getProperty foo --expand false. The value "hello ${bar}" will be returned.
```

Properties are expanded by default when you use `getProperty` or `getProperties`.

If the value of a property contains `"${}"` but you do not want it to be interpreted as a property reference, you can use the `expandable` option:

```
ectool setProperty symbols '${!@}#' --expandable false.
```

This option can be toggled in the web UI as well. Properties are expandable by default.

Because you cannot control where your expandable property might be referenced (and therefore which context is used during expansion), Electric Cloud recommends using absolute paths when referencing a property from the value of another property.

In the example above, if you define both "foo" and "bar" as properties on a project "proj1", you might assume there is no problem with the value of "foo". However, if you later reference "foo" from a job under "proj1" (for example, `$/myProject/foo`), foo will be referenced with the job step as its context. Therefore, when the value of "foo" is expanded, you will get a `PROPERTY_REFERENCE_ERROR` because "bar" is not defined in the context of the job step.

Custom property names and values

The following properties are used by the standard ElectricFlowUI, so you should use these property names whenever possible, and avoid using these names in ways that conflict with the definitions below.

- **preSummary** - if this property exists, its value is displayed in the "Status" field (on the Job Details page) for this step. This property appears *before* whatever would normally be displayed for status. If the property contains multiple lines separated by newline characters, each line is displayed on a separate line in the status field.
- **postSummary** - if this property exists, its value is displayed in the "Status" field (on the Job Details page) for this step. This property appears *after* whatever would normally be displayed for status. If the property contains multiple lines separated by newline characters, each line is displayed on a separate line in the status field.
- **summary** - if this property exists, its value is displayed in the "Status" field (in the job reports) for this step, replacing whatever would normally be displayed for status. If the property contains multiple lines separated by newline characters, each line is displayed on a separate line in the status field.

The property hierarchy

All ElectricFlow properties fall into a hierarchical structure, and you can reference any property using an absolute path from the root of the hierarchy. This includes properties you define and intrinsic properties defined by ElectricFlow.

For example, each step contains a property "resource" that provides the resource name to use for that step. All steps of a procedure exist as property sheets underneath the procedure. All procedures in a project exist as property sheets underneath the project, and so on.

The examples below illustrate some nesting relationships between objects.

For example, the notation "*project/procedures[procedureName]*" means each project object contains a property sheet named "procedures" holding all procedures defined within that project. Within the procedures property sheet, there is a nested property sheet for each procedure, named after the procedure. Thus, the name "*projects[a]/procedures[b]*" refers to a procedure named "b" contained in a project named "a."

- Each project contains procedures, schedules, credential definitions, workflow definitions, and workflows:

```
project/procedures[procedureName]
project/schedules[scheduleName]
project/credentials[credentialName]
project/workflowDefinitions[workflowName]
project/workflows[workflowName]
```

- Each procedure contains steps and parameters. The parameters are "formal parameters," meaning they specify parameter names the procedure will accept, whether each parameter is required, and so on:

```
procedure/steps[stepName]
procedure/formalParameters[paramenterName]
```

- Steps that invoke subprocedures contain parameter values for the subprocedure. These are called "actual parameters" because they provide actual values that will be passed into the subprocedure:

```
step/actualParameters[parameterName]
```

- Each job contains a collection of job step objects for steps in the outermost procedure, along with parameter values passed into the job when it was invoked:

```
job/jobSteps[stepName]
job/actualParameters[parameterName]
```

- If a job step invokes a nested subprocedure, its property sheet contains parameter values that were passed into the nested subprocedure, plus all job steps corresponding to that procedure:

```
jobStep/actualParameters[parameterName]
jobStep/jobSteps[stepName]
```

- Schedules contain parameter values for the procedures they invoke. These are called "actual parameters" because they provide actual values passed into the procedure:

```
schedule/actualParameters[parameterName]
```

- Workflow definitions contain state definitions:

```
workflowDefinition/stateDefinitions[stateDefinitionName]
```

- Transition definitions contain actual parameters:

```
transitionDefinition/actualParameters[parameterName]
```

- State definitions contain transition definitions, actual parameters, and formal parameters:
stateDefinition/actualParameters[parameterName]
stateDefinition/formalParameters[parameterName]
stateDefinition/transitionDefinitions[transitionDefinitionName]
- Workflows contains states:
workflow/states[stateName]
- Transitions contain actual parameters:
transition/actualParameters[parameterName]
- States contain transitions, actual parameters, and formal parameters:
state/actualParameters[parameterName]
state/formalParameters[parameterName]
state/transitions[transitionName]

Special property references

ElectricFlow also supports several special property reference forms that are described in the following subsections.

increment

Use this form to increment the value of an integer property before returning its value. For example, suppose property xyz has the value 43. The property reference `$/increment xyz` first increments the value of property xyz to 44, then returns 44.

timestamp

Use this form to generate a formatted timestamp value. For example, the property reference `$/timestamp yyyy-MM-d hh:mm` returns the current time in a form such as "2007-Jun-19 04:36". The pattern following `/timestamp` specifies how to format the time and defaults to "yyyyMMddHHmm". The pattern follows conventions for the Java class `SimpleDateFormat`, where various letters are substituted with various current time elements. For more information, go to <http://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html> for `SimpleDateFormat` class documentation.

Here are some of the supported substitutions:

- y - Year, such as "2007" or "07"
- M - Month, such as "April", "Apr", or "04"
- w - Week in year, such as "27"
- W - Week in month, such as "3"
- D - Day in year, such as "194"
- d - Day in month, such as "18"
- E - Day in the week, such as "Tuesday" or "Tue"
- a - am/pm marker, such as "PM"
- H - Hour in day (0-23), such as "7"

h - Hour in am/pm (1-12), such as "11"

m - Minute in hour, such as "30"

s - Second in minute, such as "15"

S - Millisecond, such as "908"

z - Time zone such as "Pacific Standard Time", "PST", or "GMT-08:00"

Z - Time zone such as "-0800"

Repeated letters cause longer forms to be substituted. For example, "yy" substitutes the last 2 digits of the current year, whereas "yyyy" substitutes the 4-digit year number. Single quotes can be used to substitute text directly without the above interpretations.

javascript

This form executes a Javascript code fragment inside the ElectricFlow server and returns the result computed by that code.

For example, `$/javascript 4*2` returns "8.0". Javascript code can be arbitrarily long and include multiple statements. The value of the last statement is returned by the property reference.

Javascript code executes in an interpreter that provides access to ElectricFlow properties:

1. The normal way to access property values is through Javascript objects. Global Javascript objects named "server," "projects," and so on, exist and correspond to all top-level objects in an absolute property path.
For example, there is a Javascript object named `server` that corresponds to the path `/server`, and a Javascript object named `myJob` that corresponds to the path `/myJob`. You can use either `"."` or `"["` notation to access properties within the object.
For a more complete list of top-level objects, see the ["Absolute property paths"](#) section.

Examples:

```
$/javascript myJob.mightExist]
this is a safe way to refer to an optional parameter
```

```
$/javascript (myJobStep.errors > 0) ? "step failed" : "no errors"]
test a value and return another string based on the result
```

```
$/javascript server.settings.ipAddress]
refer to a property in a nested property sheet
```

```
$/javascript server["Electric Cloud"].installDirectory]
if the property name has a space, use [""] notation
```

```
$/javascript server["Electric Cloud"]["installDirectory"]]
do not use the "." in front of the "["
```

2. You can also call the function, "getProperty". It takes a property name as argument and returns the value of that property. The case where this is most useful is when you want to access another special property reference.

Examples:

```
$/javascript getProperty ("/timestamp yyyy-MMM-d hh:mm")]  
returns the current time
```

```
$/javascript getProperty ("/increment /myJob/jsCount")]  
this is the only way to change a property via javascript
```

For example, consider the following property reference:

```
$/javascript (getProperty("/myJobStep/errors") > 0) ? "step failed" : "no  
errors"]
```

This example returns "step failed" if the property "errors" on the current job step had a value greater than zero, and it returns "no errors" otherwise.

3. If calling the "setProperty" function, similar to `getProperty`, there are two variations on the function:
 - A *global* function variation that uses the current context object.
The global function takes 2 or 3 arguments. The 3-argument version takes a context object, a path, and a value. The 2-argument version omits the context object and uses the current context object (for example, *job step*).

Example:

```
setProperty (myProject, "foo", "bar")  
would set the value of the "foo" property on the current project to "bar".
```

- An *object* function variation that can be called on objects, which uses that object as a context object. The object function takes two arguments: a "path" and a "value".

Example:

```
server.setProperty ("foo", "bar")  
would set the "foo" property on the server object to the value "var".
```

ElectricFlow Intrinsic Properties listed by object type

Each of the ElectricFlow object types [in the list below] links to a table for that object.

Each table includes a list of intrinsic properties applicable to that object and provides the property type and description.

acl	repository
artifact	resource

```

artifactVersion    resourcePool
credential          resourceUsage
directoryProvider  schedule
emailConfig        server
emailNotifier      state
formalParameter    stateDefinition
gateways           step
group              transition
job                transitionDefinition
jobStep            user
logEntry           workflow
procedure          workflowDefinition
project            workspace
property           zones
propertySheet

```

ElectricFlow Property Type definitions

This table provides Property Type definitions for each property listed in the following tables.
(A Property Type precedes each property description in the Description column.)

Type	Definition
boolean	One of two possible values - either <i>true</i> or <i>false</i> . These values are more frequently represented by the numbers "0" and "1", where "0" equals false and "1" equals true.
date	A millisecond precision UTC date in ISO 8601 form: [YYYY] - [MM] - [DD] T [hh] : [mm] Z For example, 2007-06-19T04:36:22.000Z
id	Each time an object is created, ElectricFlow generates a unique ID number for that object.

Type	Definition
name	This is a unicode string value with a maximum of 255 characters.
number	This is a simple integer numeric value.
reference	This property refers to another object.
string	This is a unicode string value with a maximum CLOB size of the database, but only the first 450 characters are indexed, which means a defined search will not "see" beyond the first 450 characters.

ElectricFlow Object / Property tables

In the following tables, the Description column displays the property type preceding the property description.

Object Type: <code>acl</code>	
Description: An <i>acl</i> is an Access Control List.	
Property Name	Description
<code>aclId</code>	<code>id</code> : The unique identifier for this <code>acl</code> object. Other objects can refer to this <code>acl</code> by its ID.
<code>inheriting</code>	<code>boolean</code> : If true, the ACL inherits ACEs from the ACL's parent.

Object Type: <code>acl</code> Description: An <i>acl</i> is an Access Control List.																													
Property Name	Description																												
<code>ownerType</code>	<p>This is any type of object the ACL controls and can be any of the objects listed below.</p> <table> <tr> <td><code>ace</code></td><td><code>project</code></td></tr> <tr> <td><code>acl</code></td><td><code>property</code></td></tr> <tr> <td><code>admin</code></td><td><code>propertySheet</code></td></tr> <tr> <td><code>artifact</code></td><td><code>repository</code></td></tr> <tr> <td><code>artifactVersion</code></td><td><code>resource</code></td></tr> <tr> <td><code>event</code></td><td><code>resourcePool</code></td></tr> <tr> <td><code>formalParameter</code></td><td><code>server</code></td></tr> <tr> <td><code>group</code></td><td><code>schedule</code></td></tr> <tr> <td><code>job</code></td><td><code>systemObject</code></td></tr> <tr> <td><code>jobStep</code></td><td><code>user</code></td></tr> <tr> <td><code>license</code></td><td><code>userSettings</code></td></tr> <tr> <td><code>notifier</code></td><td><code>workspace</code></td></tr> <tr> <td><code>procedure</code></td><td><code>plugin</code></td></tr> <tr> <td><code>procedureStep</code></td><td></td></tr> </table>	<code>ace</code>	<code>project</code>	<code>acl</code>	<code>property</code>	<code>admin</code>	<code>propertySheet</code>	<code>artifact</code>	<code>repository</code>	<code>artifactVersion</code>	<code>resource</code>	<code>event</code>	<code>resourcePool</code>	<code>formalParameter</code>	<code>server</code>	<code>group</code>	<code>schedule</code>	<code>job</code>	<code>systemObject</code>	<code>jobStep</code>	<code>user</code>	<code>license</code>	<code>userSettings</code>	<code>notifier</code>	<code>workspace</code>	<code>procedure</code>	<code>plugin</code>	<code>procedureStep</code>	
<code>ace</code>	<code>project</code>																												
<code>acl</code>	<code>property</code>																												
<code>admin</code>	<code>propertySheet</code>																												
<code>artifact</code>	<code>repository</code>																												
<code>artifactVersion</code>	<code>resource</code>																												
<code>event</code>	<code>resourcePool</code>																												
<code>formalParameter</code>	<code>server</code>																												
<code>group</code>	<code>schedule</code>																												
<code>job</code>	<code>systemObject</code>																												
<code>jobStep</code>	<code>user</code>																												
<code>license</code>	<code>userSettings</code>																												
<code>notifier</code>	<code>workspace</code>																												
<code>procedure</code>	<code>plugin</code>																												
<code>procedureStep</code>																													
<code>parentId</code>	<code>id</code> : The parent ACL.																												

[\[back to top\]](#)

Object Type: <code>artifact</code> Description: An <i>artifact</i> is an object that contains zero or more artifact versions. An artifact has two purposes: <ol style="list-style-type: none"> 1. To group artifact versions and provide a template for naming the versions 2. To restrict who can publish artifact versions, based on <code>groupId:artifactKey</code> 	
Property Name	Description
<code>acl</code>	<code>reference:acl</code>
<code>artifactId</code>	<code>id</code> : The artifact's ID number.

Object Type: artifact

Description: An *artifact* is an object that contains zero or more artifact versions.

An artifact has two purposes:

1. To group artifact versions and provide a template for naming the versions
2. To restrict who can publish artifact versions, based on `groupId:artifactKey`

Property Name	Description
artifactKey	<code>string</code> : User-specified identifier for this artifact. This field may consist of alphanumeric characters, spaces, underscores, hyphens, and periods.
artifactName	<code>name</code> : The name of this artifact.
artifactVersionNameTemplate	<code>name</code> : The template for artifact version names published to this artifact.
createTime	<code>date</code> : The time when this object was created.
description	<code>string</code> : A user-specified text description of the object.
groupId	<code>id</code> : A user-generated group name for this artifact. This field may consist of alphanumeric characters, spaces, underscores, hyphens, and periods.
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.
propertySheet	<code>reference</code> : <code>propertySheet</code>

[\[back to top\]](#)

Object Type: artifactVersion

Description: An artifact version is an object that represents a user-defined unit of related files typically produced by one job and consumed by one or more other jobs.

Property Name	Description
acl	<code>reference: acl</code>
artifactKey	<code>string</code> : User-specified identifier for this artifact. This field may consist of alphanumeric characters, spaces, underscores, hyphens, and periods.
artifactName	<code>name</code> : The name of the artifact.
artifactVersionId	<code>id</code> : The ElectricFlow-generated ID number for this artifact version.
artifactVersionName	<code>name</code> : The name of the artifact version.
artifactVersionState	<code>string</code> : Possible values are: available publishing unavailable
buildNumber	<code>number</code> : User-defined build number component of the version attribute for the artifact version.
createTime	<code>date</code> : The time when this object was created.
description	<code>string</code> : A user-specified text description of the object.
groupId	<code>id</code> : A user-generated group name for this artifact. This field may consist of alphanumeric characters, spaces, underscores, hyphens, and periods.
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
majorMinorPatch	<code>string</code> : <code>major.minor.patch</code> component of the version attribute for the artifact.
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.

Object Type: artifactVersion

Description: An artifact version is an object that represents a user-defined unit of related files typically produced by one job and consumed by one or more other jobs.

Property Name	Description
propertySheet	<code>reference</code> : <code>propertySheet</code>
publisherJobId	<code>id</code> : The ElectricFlow-generated ID number for the job that published the artifact version.
publisherJobName	<code>string</code> : The name of the job that published the artifact version.
publisherJobStepId	<code>id</code> : The ElectricFlow-generated ID number for the job step that published the artifact version.
qualifier	<code>string</code> : User-defined qualifier component of the version attribute for the artifact.
repositoryName	<code>name</code> : The name of the artifact repository.
version	<code>string</code> : An artifact version specification uses the following form: <i>major.minor.patch.qualifier.buildNumber</i>

[\[back to top\]](#)

Object Type: credential

Description: In ElectricFlow, a *credential* is an object that stores a username and password for later use.

Property Name	Description
acl	<code>reference</code> : <code>acl</code>
createTime	<code>date</code> : The time when this object was created.
credentialId	<code>id</code> : The credential's ID number.
credentialName	<code>name</code> : The name of this credential.

Object Type: credential

Description: In ElectricFlow, a *credential* is an object that stores a username and password for later use.

Property Name	Description
description	<code>string</code> : A user-specified text description of the object.
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.
password	<code>string</code> : The password corresponding to the user name and this credential.
projectName	<code>name</code> : The name of the <code>project</code> that contains this credential.
propertySheet	<code>reference</code> : <code>propertySheet</code>
userName	<code>name</code> : A saved string that represents the name portion of a credential, typically a user account name.

[\[back to top\]](#)

Object Type: directoryProvider

Description: A `directoryProvider` is an object containing information about an external directory service (LDAP or ActiveDirectory).

Property Name	Description
acl	<code>reference</code> : <code>acl</code>
commonGroupNameAttribute	<code>string</code> : The attribute in a group record that contains the common group name. If specified, this name is used only when searching for groups from an external provider.

Object Type: directoryProvider

Description: A directoryProvider is an object containing information about an external directory service (LDAP or ActiveDirectory).

Property Name	Description
createTime	<code>date</code> : The time when this object was created.
description	<code>string</code> : A user-specified text description of the object.
directoryProviderId	<code>id</code> : The ID of the directory provider.
domainName	<code>string</code> : The domain name from which Active Directory servers are automatically discovered.
emailAttribute	<code>string</code> : The attribute in a user record that contains the user's email address. If the attribute was not specified, the account name and domain name are concatenated to form an email address.
enableGroups	<code>boolean</code> : Determines whether or not external groups are enabled for the directory provider. Defaults to "true".
fullUserNameAttribute	<code>name</code> : The attribute in a user record that contains the user's full name (first and last) for display in the UI.
groupBase	<code>string</code> : This string is prepended to the <code>basedn</code> to construct the directory DN that contains group records.
groupMemberAttributes	<code>string</code> : A comma-separated attribute name list that identifies a group member.
groupMemberFilter	<code>string</code> : Two common forms of group record in LDAP directories: POSIX style groups where members are identified by account name, and <code>groupOfNames</code> or <code>uniqueGroupOfNames</code> records where members are identified by the full user DN. Both forms are supported, so the query is passed to parameters: "{0}" is replaced with the full user record DN, and "{1}" is replaced with the user's account name.
groupNameAttribute	<code>name</code> : The group record that contains the name of the group.

Object Type: `directoryProvider`

Description: A `directoryProvider` is an object containing information about an external directory service (LDAP or ActiveDirectory).

Property Name	Description
<code>groupSearchFilter</code>	<code>string</code> : This LDAP query is performed in the context of the groups directory to enumerate group records.
<code>lastModifiedBy</code>	<code>name</code> : This shows who (generally, a user name) last modified this object.
<code>managerDn</code>	<code>name</code> : The DN of a user who has read-only access to LDAP user and group directories.
<code>modifyTime</code>	<code>date</code> : The time when this object was last modified.
<code>owner</code>	<code>name</code> : The person (user name) who created the object.
<code>propertySheet</code>	<code>reference</code> : <code>propertySheet</code>
<code>providerIndex</code>	<code>number</code> : The index that specifies the search order across multiple directory providers. For example: 2 LDAP providers, one with index "0" and one with index "1" means the providers will be searched in that numerical order.
<code>providerName</code>	<code>name</code> : This human-readable name will be displayed in the user interface to identify users and groups that come from this provider.
<code>providerType</code>	<code>string</code> : <code><ldap activedirectory></code>
<code>realm</code>	<code>string</code> : An identifier (string) used for LDAP directory providers so users and groups (within LDAP) can be uniquely identified in "same name" collisions across multiple directory providers. The realm is appended to the user or group name when stored in the ElectricFlow server. For example, <code><user>@dir</code> (where the realm is set to "dir").
<code>url</code>	<code>string</code> : The server URL is in the form <code>protocol://host:port/basedn</code> . Protocol is either <code>ldap</code> or <code>ldaps</code> (for secure LDAP).

Object Type: directoryProvider

Description: A directoryProvider is an object containing information about an external directory service (LDAP or ActiveDirectory).

Property Name	Description
userBase	string: This string is prepended to the <code>basedn</code> to construct the directory DN that contains user records.
userNameAttribute	name: The attribute in a user record that contains the user's account name.
userSearchFilter	string: This LDAP query is performed in the context of the user directory to search for a user by account name. The string "{0}" is replaced with the user's login ID. Typically, the query compares a user record attribute with the substituted user login ID.
userSearchSubtree	boolean: If true, the subtree below the user base is searched recursively.
useSSL	boolean: This flag is used to specify SSL to communicate with your Active Directory servers. Note: Transport Layer Security (TLS) has replaced Secure Sockets Layer version 3.0 (SSLv3) on the ElectricFlow web server and the ElectricFlow server.

[\[back to top\]](#)

Object Type: emailConfig

Description: An *emailConfig* is an object that stores information created and used to communicate with the email server.

Property Name	Description
acl	reference: <code>acl</code>
configName	string: The name of the email configuration.

Object Type: emailConfig

Description: An *emailConfig* is an object that stores information created and used to communicate with the email server.

Property Name	Description
createTime	date : The time when this object was created.
description	string : A user-specified text description of the object.
emailConfigId	id : The ElectricFlow-generated ID for the email configuration.
emailConfigName	string : The name of the email configuration.
lastModifiedBy	name : This shows who (generally, a user name) last modified this object.
mailFrom	string : The email address used as the email sender address for notifications.
mailHost	string : The name of the email server host.
mailPort	number : The port number for the mail server. The protocol software determines the default value (25 for SMTP and 465 for SSMTP).
mailProtocol	string : This is either SSMTP or SMTP (not case sensitive). Default is SMTP.
mailUser	name : An individual or a generic name like "ElectricFlow" -- the name of the email user on whose behalf ElectricFlow sends email notifications.
modifyTime	date : The time when this object was last modified.
owner	name : The person (user name) who created the object.
propertySheet	reference : propertySheet

[\[back to top\]](#)

Object Type: emailNotifier

Description: An *emailNotifier* is an object that stores information created and used to notify users about various types information, including status.

Property Name	Description
acl	<code>reference:</code> <code>acl</code>
condition	<code>string</code> : Only send mail if the condition evaluates to "true". The condition is a string subject to property expansion. The notification will NOT be sent if the expanded string is "false" or "0". If no condition is specified, the notification is ALWAYS sent.
configName	<code>string</code> : The name of the email configuration.
container	<code>id</code> : The object ID to which the email notifier is attached (for example: procedure-123).
containerType	<code>string</code> : The type of object that the notifier is attached to (procedure, step, job, jobstep).
createTime	<code>date</code> : The time when this object was created.
description	<code>string</code> : A user-specified text description of the object.
destinations	<code>string</code> : A space-separated list of valid email addresses, email aliases, or ElectricFlow user names, or a string subject to property expansion that expands into such a list.
emailNotifierId	<code>id</code> : The ElectricFlow-generated ID for the email notifier.
eventType	<code>string</code> : <code><onEnter onStart onCompletion></code> <code>onStart</code> triggers an email notification when the job or job step begins. <code>onCompletion</code> , default, triggers an email notification when the job finishes, no matter how it finishes.
formattingTemplate	<code>string</code> : The email address used as the email sender address for notifications.

Object Type: emailNotifier

Description: An *emailNotifier* is an object that stores information created and used to notify users about various types information, including status.

Property Name	Description
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
modifyTime	<code>date</code> : The time when this object was last modified.
notifierName	<code>name</code> : The name of the email notifier.
owner	<code>name</code> : The person (user name) who created the object.
propertySheet	<code>reference</code> : <code>propertySheet</code>

[\[back to top\]](#)

Object Type: formalParameter

Description: A *formalParameter* is a parameter expected by a procedure, including its name, a default value, and an indication of whether the parameter is required. Formal parameters are different from "actual parameters"--- formal parameters define the type of parameters a procedure is expecting, and actual parameters provide values to use at run-time.

Property Name	Description
container	<code>string</code> : An object ID for a "container" that contains formal parameters.
containerType	<code>string</code> : The type of object containing the formal parameter.
createTime	<code>date</code> : The time when this object was created.
defaultValue	<code>string</code> : The <code>formalParameter</code> 's default value.

Object Type: `formalParameter`

Description: A *formalParameter* is a parameter expected by a procedure, including its name, a default value, and an indication of whether the parameter is required. Formal parameters are different from "actual parameters"--- formal parameters define the type of parameters a procedure is expecting, and actual parameters provide values to use at run-time.

Property Name	Description
<code>description</code>	<code>string</code> : A user-specified text description of the object.
<code>expansionDeferred</code>	<code>boolean</code> : Determines whether or not the property expansion is deferred.
<code>formalParameterId</code>	<code>id</code> : This is this formal parameter's ID.
<code>formalParameterName</code>	<code>name</code> : This is this formal parameter's name.
<code>lastModifiedBy</code>	<code>name</code> : This shows who (generally, a user name) last modified this object.
<code>modifyTime</code>	<code>date</code> : The time when this object was last modified.
<code>owner</code>	<code>name</code> : The person (user name) who created the object.
<code>required</code>	<code>boolean</code> : If true, a value for this parameter must be supplied when the procedure is called.
<code>type</code>	<code>name</code> : The custom type of <code>formalParameter</code> .

[\[back to top\]](#)

Object Type: gateway

Description: To communicate with a resource, workspace, or artifact repository server in another zone, a "gateway" must be created. A gateway object contains two resource (agent) machines, for example, GatewayResource1 and GatewayResource2—each configured to communicate with the other. One gateway resource resides in the *source* zone and the other in the *target* zone. A gateway is bidirectional and informs the ElectricFlow server that each gateway machine is configured to communicate with its other gateway machine (in another zone).

Property Name	Description
acl	<code>reference</code> : acl
createTime	<code>date</code> : The time when this object was created.
description	<code>string</code> : A user-specified text description of the object.
gatewayDisabled	<code>boolean</code> : If set to 1 (true), the gateway is disabled.
gatewayId	<code>id</code> : The unique ElectricFlow-generated ID for this gateway.
gatewayName	<code>string</code> : The name of the gateway.
hostName1	<code>string</code> : The agent host name where <i>Resource1</i> resides. This host name is used by <i>Resource2</i> to communicate with <i>Resource1</i> . Do not specify this option if you want to use the host name from <i>Resource1</i> 's definition.
hostName2	<code>string</code> : The agent host name where <i>Resource2</i> resides. This host name is used by <i>Resource1</i> to communicate with <i>Resource2</i> . Do not specify this option if you want to use the host name from <i>Resource2</i> 's definition.
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.
port1	<code>number</code> : The port number used by <i>Resource1</i> - defaults to the port number used by the resource.

Object Type: gateway

Description: To communicate with a resource, workspace, or artifact repository server in another zone, a "gateway" must be created. A gateway object contains two resource (agent) machines, for example, GatewayResource1 and GatewayResource2—each configured to communicate with the other. One gateway resource resides in the *source* zone and the other in the *target* zone. A gateway is bidirectional and informs the ElectricFlow server that each gateway machine is configured to communicate with its other gateway machine (in another zone).

Property Name	Description
port2	<code>number</code> : The port number used by <i>Resource2</i> - defaults to the port number used by the resource.
propertySheet	<code>reference</code> : <code>propertySheet</code>
resourceName1	<code>string</code> : The name of your choice for the first of two required gateway resources. Do not include "spaces" in a resource name.
resourceName2	<code>string</code> : The name of your choice for the second of two required gateway resources. Do not include "spaces" in a resource name.

[\[back to top\]](#)**Object Type: group**

Description: This is any *group* of users known to the ElectricFlow server, including groups defined locally within the server and those groups defined in external repositories such as LDAP or ActiveDirectory.

Property Name	Description
acl	<code>reference</code> : <code>acl</code>
createTime	<code>date</code> : The time when this object was created.
groupId	<code>id</code> : The unique ElectricFlow-generated group ID.
groupName	<code>name</code> : This is this group's name.

Object Type: group

Description: This is any *group* of users known to the ElectricFlow server, including groups defined locally within the server and those groups defined in external repositories such as LDAP or ActiveDirectory.

Property Name	Description
<code>lastModifiedBy</code>	<code>name</code> : This shows who (generally, a user name) last modified this object.
<code>modifyTime</code>	<code>date</code> : The time when this object was last modified.
<code>mutable</code>	<code>boolean</code> : If true, the member list of this group is editable within ElectricFlow via the web UI or the <code>modifyGroup</code> API.
<code>owner</code>	<code>name</code> : The person (user name) who created the object.
<code>propertySheet</code>	<code>reference</code> : <code>propertySheet</code>
<code>providerName</code>	<code>name</code> : The name of the <code>directory provider</code> that controls this group.

[\[back to top\]](#)

Object Type: job

Description: A *job* is an ElectricFlow structure associated with invoking a procedure. A new job is created each time a procedure begins to execute. The job keeps track of all data associated with the procedure's execution, such as the running time of each step and any errors that may occur during the step. ElectricFlow retains job information after the job completes so you can examine what occurred.

Property Name	Description
<code>abortStatus</code>	<code>string</code> : If set, indicates the step was aborted. Values will be either <code>ABORT</code> or <code>FORCE_ABORT</code> - indicating how the step was aborted.
<code>abortedBy</code>	<code>name</code> : This is the user who issued the "abort."

Object Type: `job`

Description: A *job* is an ElectricFlow structure associated with invoking a procedure. A new job is created each time a procedure begins to execute. The job keeps track of all data associated with the procedure's execution, such as the running time of each step and any errors that may occur during the step. ElectricFlow retains job information after the job completes so you can examine what occurred.

Property Name	Description
<code>acl</code>	<code>reference</code> : <code>acl</code>
<code>actualParameter</code>	<code>reference</code> : <code>propertySheet</code> An <code>actualParameter</code> is an object that provides the value for a parameter, which is passed to a procedure when it is invoked. Actual parameters can be provided for jobs and nested subprocedures within a job. Actual parameters are different from "formal parameters"- formal parameters define the parameters a procedure is expecting, and actual parameters provide values to use at run-time.
<code>callingState</code>	<code>string</code> : The full property path to the "calling state", which can appear on <code>subjobs</code> and <code>subworkflows</code> of a workflow.
<code>callingStateId</code>	<code>id</code> : The ElectricFlow-generated ID number for the calling state.
<code>combinedStatus</code>	Provides more inclusive step status output - the resulting query output may contain up to three sub-elements: <code>status message properties</code>
<code>createTime</code>	<code>date</code> : The time when this object was created.
<code>credentialName</code>	<code>name</code> : The name of the <code>credential</code> being used for impersonation when the job runs commands on a resource.
<code>deleted</code>	The object was marked for background deletion. Possible values are "0" or "1". Default is "0" (not set).
<code>directoryName</code>	<code>name</code> : The name of this job's directory within each workspace for this job.

Object Type: `job`

Description: A *job* is an ElectricFlow structure associated with invoking a procedure. A new job is created each time a procedure begins to execute. The job keeps track of all data associated with the procedure's execution, such as the running time of each step and any errors that may occur during the step. ElectricFlow retains job information after the job completes so you can examine what occurred.

Property Name	Description
<code>elapsedTime</code>	<code>number</code> : The number of milliseconds between the start and end times for the job.
<code>errorCode</code>	<code>errorCode</code> : When the outcome is <code>error</code> , this property displays the error code, identifying which error occurred.
<code>errorMessage</code>	<code>string</code> : When the outcome is <code>error</code> , this property displays the error description.
<code>external</code>	<code>boolean</code> : If "true", the job is external.
<code>finish</code>	<code>date</code> : The time this job completed.
<code>jobId</code>	<code>id</code> : This is this job's ID number, which is a UUID.
<code>jobName</code>	<code>name</code> : This is this job's name.
<code>lastModifiedBy</code>	<code>name</code> : This shows who (generally, a user name) last modified this object.
<code>launchedByUser</code>	<code>name</code> : The name of the user or project principal that explicitly launched the job. This property is blank when the job is launched by a schedule.
<code>licenseWaitTime</code>	The sum of time all job steps had to wait for a license.
<code>liveProcedure</code>	<code>name</code> : The current procedure name from which this job was created – if the procedure was renamed since the job launched, this will be the procedure's new name, and if the procedure was deleted, this will be null.

Object Type: job

Description: A *job* is an ElectricFlow structure associated with invoking a procedure. A new job is created each time a procedure begins to execute. The job keeps track of all data associated with the procedure's execution, such as the running time of each step and any errors that may occur during the step. ElectricFlow retains job information after the job completes so you can examine what occurred.

Property Name	Description
liveSchedule	<code>name</code> : The current schedule name that launched this job – if the schedule was renamed since the job was launched, this will be the schedule's new name, and if the schedule was deleted, this will be null.
modifyTime	<code>date</code> : The time when this object was last modified.
outcome	The overall result of the job: success, warning, error, or skipped.
owner	<code>name</code> : The person (user name) who created the object.
priority	Values can be low, normal (default), high, or highest.
procedureName	<code>name</code> : The name of the procedure that defines the job's steps.
projectName	<code>name</code> : The name of the project that contains this job.
propertySheetId	<code>reference</code> : propertySheet
resourceWaitTime	The sum of time all job steps had to wait for a resource. This could indicate that eligible resources for the step have reached their step limit, are in-use but the step requires a resource exclusively, or resources are down.
runAsUser	<code>name</code> : The name of the user being impersonated in this job.
scheduleName	<code>name</code> : The schedule name that launched this job – this field differs from <code>liveSchedule</code> in that it is written at job creation time only, and not changed, even if the schedule is renamed or deleted.
start	<code>date</code> : The time when this job began executing.

Object Type: `job`

Description: A *job* is an ElectricFlow structure associated with invoking a procedure. A new job is created each time a procedure begins to execute. The job keeps track of all data associated with the procedure's execution, such as the running time of each step and any errors that may occur during the step. ElectricFlow retains job information after the job completes so you can examine what occurred.

Property Name	Description
<code>status</code>	<p>Possible values are:</p> <p><code>pending</code> - the job was created, but it is waiting for prerequisite steps to complete</p> <p><code>runnable</code> - the job step is waiting for a resource</p> <p><code>scheduled</code> - the job was assigned a resource, but the command has not started running</p> <p><code>running</code> - the job/job step is running a command on the assigned resource</p> <p><code>completed</code> - the job/job step has completed</p>
<code>totalWaitTime</code>	The total sum of license, resource, a precondition, and workspace wait times job steps were restricted and had to wait to process.
<code>workspaceWaitTime</code>	The sum of time all job steps had to wait for a workspace.

[\[back to top\]](#)

Object Type: `jobStep`

Description: A *jobStep* is an instance of a step that ran - a single invocation of a command on a resource or a single call to a subprocedure within the context of a [job](#).

Property Name	Description
<code>abortStatus</code>	<code>string</code> : If set, indicates the step was aborted. Values will be either <code>ABORT</code> or <code>FORCE_ABORT</code> - indicating how the step was aborted.
<code>abortedBy</code>	<code>name</code> : The user who issued the "abort."

Object Type: `jobStep`

Description: A *jobStep* is an instance of a step that ran - a single invocation of a command on a resource or a single call to a subprocedure within the context of a [job](#).

Property Name	Description
<code>acl</code>	<code>reference</code> : acl
<code>actualParameter</code>	<code>reference</code> : propertySheet An <code>actualParameter</code> is an object that provides the value for a parameter, which is passed to a procedure when it is invoked. Actual parameters can be provided for jobs and nested subprocedures within a job. Actual parameters are different from "formal parameters"- formal parameters define the parameters a procedure is expecting, and actual parameters provide values to use at run-time.
<code>alwaysRun</code>	<code>boolean</code> : If true, this step runs even if the job is aborted before the step completes. Note that a "force abort" will abort an <code>alwaysRun</code> step.
<code>assignedResourceName</code>	<code>name</code> : The name of the resource assigned to this step by the step scheduler.
<code>broadcast</code>	<code>boolean</code> : If true, this step runs on all resources in a pool.
<code>combinedStatus</code>	Provides more inclusive step status output - the resulting query output may contain up to three sub-elements: <code>status message properties</code>
<code>command</code>	<code>string</code> : This property specifies the command this step runs.
<code>condition</code>	<code>string</code> : If this element is not present, the event is ALWAYS triggered. If specified, the event is triggered only if the value of the condition argument is TRUE; if not true, a boolean value of "false" or "0" was used. Condition arguments can be a literal, a fixed value string, or a string subject to property expansion.
<code>conditionExpanded</code>	<code>boolean</code> : The result of the expansion on the step condition.
<code>createTime</code>	<code>date</code> : The time when this object was created.

Object Type: `jobStep`

Description: A *jobStep* is an instance of a step that ran - a single invocation of a command on a resource or a single call to a subprocedure within the context of a [job](#).

Property Name	Description
<code>delayUntil</code>	For a step that was rescheduled due to a resource or workspace problem, this is the next time the step will be eligible to run.
<code>elapsedTime</code>	number : The number of milliseconds between the start and end times for the <code>jobStep</code> .
<code>errorCode</code>	errorCode : This property appears when the outcome is <code>error</code> and displays the error code, identifying which error occurred.
<code>errorHandling</code>	<p>This is the error handling policy copied from the procedure step and indicates how the step responds to errors:</p> <p><code>failProcedure</code> - The current procedure continues, but the overall status is error (default).</p> <p><code>abortProcedure</code> - Aborts the current procedure, but allows already-running steps in the current procedure to complete.</p> <p><code>abortProcedureNow</code> - Aborts the current procedure and terminates running steps in the current procedure.</p> <p><code>abortJob</code> - Aborts the entire job, terminates running steps, but allows <code>alwaysRun</code> steps to run.</p> <p><code>abortJobNow</code> - Aborts the entire job and terminates all running steps, including <code>alwaysRun</code> steps.</p> <p><code>failProcedure</code> - The current procedure continues, but the overall status is error (default).</p> <p><code>ignore</code> - Continues as if the step succeeded.</p>
<code>errorMessage</code>	string : When the outcome is <code>error</code> , this property displays an error message description.
<code>exclusive</code>	boolean : If true, this step acquires and retains its resource exclusively.

Object Type: `jobStep`

Description: A *jobStep* is an instance of a step that ran - a single invocation of a command on a resource or a single call to a subprocedure within the context of a [job](#).

Property Name	Description
<code>exclusiveMode</code>	<code>string</code> : Possible values are: none, job, step, call
<code>exitCode</code>	<code>number</code> : This is this step's exit code.
<code>external</code>	<code>boolean</code> : If "true", the job is external.
<code>finish</code>	<code>date</code> : The time when this job step completed.
<code>hostName</code>	<code>name</code> : The name of the host where this step was invoked (copied from the resource)
<code>job</code>	<code>id</code> : This is the ID number of the job that owns the job step. The string representation of the job is its name, so <code>\$(job)</code> evaluates to the job name, but <code>\$(job/foo)</code> is also legal and refers to the foo property of the job.
<code>jobId</code>	<code>id</code> : This is this job's ID number, which is a UUID.
<code>jobName</code>	<code>name</code> : This is the name of this step's job.
<code>jobStepId</code>	<code>id</code> : This is this step's ID number.
<code>lastModifiedBy</code>	<code>name</code> : This shows who (generally, a user name) last modified this object.
<code>licenseWaitTime</code>	The length of time this job step had to wait to process because the license limit was reached or exceeded.
<code>liveProcedure</code>	<code>string</code> : The current procedure name for the procedure step from which the job or job step was created – if the procedure step was renamed since the job or job step was launched, this is the procedure step's new name, and if the procedure step was deleted, this will be null.

Object Type: `jobStep`

Description: A *jobStep* is an instance of a step that ran - a single invocation of a command on a resource or a single call to a subprocedure within the context of a [job](#).

Property Name	Description
<code>liveProcedureStep</code>	name: This property shows the current procedure name for the procedure step from which this job step was created – if the procedure step was renamed since the job was launched, this will be the procedure step's new name, and if the procedure step was deleted, this will be null.
<code>logFileName</code>	name: The name of the log file produced by this step, relative to the job's workspace directory.
<code>modifyTime</code>	date: The time when this object was last modified.
<code>outcome</code>	string: The overall result of the job - possible values are: <code>success</code> , <code>warning</code> , <code>error</code> , or <code>skipped</code>
<code>owner</code>	name: The person (user name) who created the object.
<code>parallel</code>	boolean: If true, this step runs in parallel with other adjacent steps also marked to run in parallel.
<code>postExitCode</code>	number: This is this step's post processor exit code.
<code>postLogFileName</code>	name: The log file name produced by this step's post processor.
<code>postProcessor</code>	string: The post processor name used to gather information about this step.
<code>precondition</code>	string: By default, if a step has no precondition, it will run when scheduled. Set this property to make a step wait until one or more dependent conditions are met. When a job step is eligible to transition from pending to runnable, a <i>precondition</i> is evaluated. A <i>precondition</i> is a fixed text or text embedding property reference that is evaluated to TRUE or FALSE. An empty string, a <code>"0"</code> or <code>"false"</code> is interpreted as FALSE. Any other result string is interpreted as TRUE. The step will block until the precondition is TRUE.

Object Type: `jobStep`

Description: A *jobStep* is an instance of a step that ran - a single invocation of a command on a resource or a single call to a subprocedure within the context of a [job](#).

Property Name	Description
<code>procedureName</code>	<code>name</code> : The name of the procedure that contains this <code>jobStep</code> .
<code>projectName</code>	<code>name</code> : The name of the project that contains this <code>jobStep</code> .
<code>propertySheet</code>	<code>reference</code> : propertySheet
<code>releaseExclusive</code>	<code>boolean</code> : A Boolean value indicating whether this step should release its resource upon completion.
<code>releaseMode</code>	<code>string</code> : Possible values are: none, release, releaseToJob
<code>resourceName</code>	<code>name</code> : The name of the resource or pool this step should use to run on.
<code>resourceSource</code>	This property indicates whether the resource for the job step is explicitly specified by the step or was defaulted from the procedure: procedure or <code>procedureStep</code> .
<code>resourceWaitTime</code>	The length of time this job step stalled because it could not get a resource. This could indicate that eligible resources for the step have reached their step limit, are in-use but the step requires a resource exclusively, or resources are down.
<code>retries</code>	<code>number</code> : This is a number - the number of retries before the request times out.
<code>runAsUser</code>	<code>name</code> : The name of the user being impersonated in this job step.
<code>runnable</code>	<code>date</code> : The time when the step became runnable.
<code>runTime</code>	<code>number</code> : The number of milliseconds the step command spent running on a resource.

Object Type: `jobStep`

Description: A *jobStep* is an instance of a step that ran - a single invocation of a command on a resource or a single call to a subprocedure within the context of a [job](#).

Property Name	Description
<code>shell</code>	name : The shell used to execute the step's commands on a resource. The script name is inserted into the command at the position of a " {0} " marker in the command, or appended as a final argument if no marker is present. A filename suffix adjacent to the marker will be appended to the script name. For more information on shells, see the Shell definition in the Step Help topic.
<code>start</code>	date : The time when this job step began executing.
<code>status</code>	<p>Possible values are:</p> <p><code>pending</code> - the job step was created, but it is waiting for prerequisite steps to complete</p> <p><code>runnable</code> - the job step is waiting for a resource</p> <p><code>scheduled</code> - the job step was assigned a resource, but the command has not started running</p> <p><code>running</code> - the job/job step is running a command on the assigned resource</p> <p><code>completed</code> - the job/job step has completed</p>
<code>stepName</code>	name : This is this step's name.
<code>subprocedure</code>	name : The nested procedure name to call when this step executes.
<code>subproject</code>	name : This property appears if the subprocedure element is present and specifies the project to which the subprocedure belongs (by default, the current project is used.)
<code>timeLimit</code>	number : The maximum length of time this step is allowed to run.
<code>timeout</code>	date : The time when this job step will be automatically aborted if it has not yet completed.

Object Type: `jobStep`

Description: A *jobStep* is an instance of a step that ran - a single invocation of a command on a resource or a single call to a subprocedure within the context of a [job](#).

Property Name	Description
<code>totalWaitTime</code>	The sum of resource, workspace, and license wait times for this job step.
<code>waitTime</code>	number : The number of milliseconds the step spent between runnable and running (for example, waiting for a resource).
<code>workingDirectory</code>	name : The name of this step's working directory.
<code>workspaceName</code>	name : The workspace name used by the <code>jobStep</code> object.
<code>workspaceWaitTime</code>	The time this job step had to wait because no workspace was found or available.

[\[back to top\]](#)

Object Type: logEntry

Description: A *logEntry* is an object containing various types of information available in a log file generated from anywhere in the ElectricFlow system.

The *Event Log* can be configured for auto-deletion. By default, Event Log retention is 30 days. The ElectricFlow server automatically marks old log entries for deletion and the background deleter cleans out old log entries.

To change the default settings, go to Administration > Server > Settings and modify the Event log retain time and the Maximum background delete delay fields

Note: Setting the "retain" period to "0" disables the automatic deletion mechanism, allowing you to use an external cleanup script to implement a custom cleanup policy.

Property Name	Description
category	(currently not used)
container	<code>string</code> : Typically, this is the type and name of the workflow or job with a corresponding ID.
containerName	<code>name</code> : The name of the container.
containerType	<code>string</code> : The type of object the log entry pertains to (for example, procedure, job step, step).
deleted	<code>byte</code> : The object was marked for background deletion. Possible values are "0" or "1". Default is "0" (which means "deleted" is not set).
logEntryId	<code>id</code> : The log entry ElectricFlow-generated ID.
message	<code>string</code> : The message text will either be informational or display the warning or error message. The message may contain important information about a resource or workspace issue.
principal	<code>string</code> : The user or project principal from the session that was active when the event occurred.

Object Type: logEntry

Description: A *logEntry* is an object containing various types of information available in a log file generated from anywhere in the ElectricFlow system.

The *Event Log* can be configured for auto-deletion. By default, Event Log retention is 30 days. The ElectricFlow server automatically marks old log entries for deletion and the background deleter cleans out old log entries.

To change the default settings, go to Administration > Server > Settings and modify the Event log retain time and the Maximum background delete delay fields

Note: Setting the "retain" period to "0" disables the automatic deletion mechanism, allowing you to use an external cleanup script to implement a custom cleanup policy.

Property Name	Description
severity	<code>string</code> : Severity can be either TRACE, DEBUG, INFO, WARN, ERROR
subject	<code>string</code> : The object associated with the message.
subjectName	<code>name</code> : The name of the object associated with the message.
subjectType	<code>string</code> : (similar to <code>container</code>) Refers to the object the event concerns. This may be the same as the container, or it may be a different object that is related to the event in some manner.
time	<code>string</code> : The time the event was logged.

[\[back to top\]](#)

Object Type: procedure

Description: A *procedure* describes a series of actions to be performed on one or more resources. A procedure contains steps and properties and can define formal parameters.

Property Name	Description
acl	<code>reference: acl</code>
createTime	<code>date</code> : The time when this object was created.
credentialName	<code>name</code> : The name of the credential assigned to this job step.
description	<code>string</code> : A user-specified text description of the object.
jobNameTemplate	<code>name</code> : The template used to name jobs created from this procedure.
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.
procedureId	<code>id</code> : This is this procedure's ID number.
procedureName	<code>name</code> : This is this procedure's name.
projectName	<code>name</code> : The name of the <code>project</code> that contains this procedure.
propertySheet	<code>reference: propertySheet</code>
resourceName	<code>name</code> : The name of the resource or pool this procedure should use to run on.
workspaceName	<code>name</code> : The workspace name used by the procedure object.

[\[back to top\]](#)

Object Type: project

Description: A *project* is an object used in ElectricFlow to organize information. A project contains procedures, schedules, credentials, and properties.

Property Name	Description
acl	<code>reference: acl</code>
createTime	<code>date</code> : The time when this object was created.
credentialName	<code>name</code> : The name of the credential assigned to this project.
deleted	The object was marked for background deletion. Possible values are "0" or "1". Default is "0" (not set).
description	<code>string</code> : A user-specified text description of the object.
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.
pluginName	<code>name</code> : The name of the plugin associated with this project.
projectId	<code>id</code> : This is this project's ID.
projectName	<code>name</code> : The name of the project.
propertySheet	<code>reference: propertySheet</code>
resourceName	<code>name</code> : The name of the resource.
workspaceName	<code>name</code> : This is the workspace name used by the project.

[\[back to top\]](#)

Object Type: property

Description: A *property* is a string value with a name. Properties can have arbitrary names and values. You can attach properties to any object in the ElectricFlow system, such as a project, procedure, or job. After a property is created, it is stored in the ElectricFlow database. You can retrieve and/or modify the value later, and you can delete properties you no longer need. Properties provide a flexible and powerful mechanism to manage data about your builds.

Note: The name "properties" is NOT a valid property name.

Property Name	Description
createTime	<code>date</code> : The time when this object was created.
description	<code>string</code> : A user-specified text description of the object.
expandable	<code>boolean</code> : If set to true, the property value will undergo string expansion when it is retrieved.
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.
canonicalPath	<code>string</code> : The path that specifies the property object.
propertyId	<code>id</code> : This property's ID number.
propertyName	<code>name</code> : This property's name.
propertySheetId	<code>reference</code> : <code>propertySheet</code> – If the property is a nested property sheet, the <code>propertySheet</code> refers to the nested sheet.
value	<code>string</code> : The property or actual parameter's value - if this property is a string property.

[\[back to top\]](#)

Object Type: `propertySheet`

Description: A property value can be a simple string or a nested *propertySheet* containing its own properties. Property sheets can be nested to any depth, which allows you to create hierarchical collections of information.

Most objects have an associated property sheet that contains "custom properties" created by user scripts.

Property Name	Description
<code>acl</code>	<code>reference</code> : <code>acl</code>
<code>createTime</code>	<code>date</code> : The time when this object was created.
<code>lastModifiedBy</code>	<code>name</code> : This shows who (generally, a user name) last modified this object.
<code>modifyTime</code>	<code>date</code> : The time when this object was last modified.
<code>owner</code>	<code>name</code> : The person (user name) who created the object.
<code>propertySheetId</code>	<code>id</code> : This property sheet's ID.

[\[back to top\]](#)

Object Type: `repository`

Description: A *repository* is an object that stores artifact versions. This object primarily contains information on how to connect to a particular artifact repository. Similar to steps in a procedure, repository objects are in a user-specified order. When retrieving artifact versions, repositories are queried in this order until one containing the desired artifact version is found.

Property Name	Description
<code>acl</code>	<code>reference</code> : <code>acl</code>
<code>createTime</code>	<code>date</code> : The time when this object was created.
<code>description</code>	<code>string</code> : A user-specified text description for this object.

Object Type: repository

Description: A *repository* is an object that stores artifact versions. This object primarily contains information on how to connect to a particular artifact repository. Similar to steps in a procedure, repository objects are in a user-specified order. When retrieving artifact versions, repositories are queried in this order until one containing the desired artifact version is found.

Property Name	Description
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.
propertySheet	<code>reference</code> : <code>propertySheet</code>
repositoryDisabled	<code>boolean</code> : Determines whether the repository is disabled. Default is "false".
repositoryId	<code>id</code> : The ElectricFlow-generated ID number of the repository.
repositoryIndex	<code>number</code> : The order of the repository, within a list of repositories.
repositoryName	<code>name</code> : The name of the repository.
url	<code>string</code> : The server URL is in the form <code>protocol://host:port/</code> . Typically, the repository server is configured to listen on port 8200 for <code>https</code> requests, so a typical URL looks like <code>https://host:8200/</code> .
zoneName	<code>name</code> : The name of the zone where this repository is or will reside.

[\[back to top\]](#)

Object Type: resource

Description: A *resource* is associated with a server machine available to ElectricFlow for running steps. A resource has a logical name in addition to a host name. A resource can be associated with one or more pools. The resource may be a "proxy target" machine that communicates with the ElectricFlow server through an ElectricFlow agent proxy machine.

Property Name	Description
acl	<code>reference</code> : acl
agentState	<code>string</code> : Specific information about an agent, including the state of the agent. Possible values are: unknown alive down
artifactCacheDirectory	<code>string</code> : The directory on the agent host where retrieved artifacts are stored.
createTime	<code>date</code> : The time when this object was created.
description	<code>string</code> : A user-specified text description of the object.
exclusiveJobId	<code>id</code> : The ID number of the job that owns this resource, which occurs when one of the job's steps requests exclusive use of the resource for the duration of the job.
exclusiveJobName	<code>name</code> : The name of the job that owns this resource, which occurs when one of the job's steps requests exclusive use of the resource for the duration of the job.
exclusiveJobStepId	<code>id</code> : The ID number of the job step that owns this resource, which occurs when one of the steps request exclusive use of the resource for the duration of the job.
exclusiveJobStepName	<code>name</code> : The name of the job step that owns this resource, which occurs when one of the steps request exclusive use of the resource for the duration of the job.
gateways	The gateway names associated with this resource.

Object Type: `resource`

Description: A *resource* is associated with a server machine available to ElectricFlow for running steps. A resource has a logical name in addition to a host name. A resource can be associated with one or more pools. The resource may be a "proxy target" machine that communicates with the ElectricFlow server through an ElectricFlow agent proxy machine.

Property Name	Description
<code>hostName</code>	<code>name</code> : The computer name or IP address for the machine containing the ElectricFlow agent for this resource.
<code>hostOS</code>	<code>string</code> : The full name of the host operating system, plus its version. However, if this host is a proxy, "proxied" appears as the host description/name.
<code>hostPlatform</code>	<code>string</code> : Examples for "platform" are: Windows, Linux, HPUX, and so on. However, if this host is a proxy, "proxied" appears as the <code>hostPlatform</code> name.
<code>lastModifiedBy</code>	<code>name</code> : This shows who (generally, a user name) last modified this object.
<code>lastRunTime</code>	<code>date</code> : The most recent time a job step ran on the resource.
<code>modifyTime</code>	<code>date</code> : The time when this object was last modified.
<code>owner</code>	<code>name</code> : The person (user name) who created the object.
<code>pools</code>	<code>string</code> : A space-separated list of one or more pool names where this resource is a member. Steps defined to run on a resource pool will run on any available member (resource) in the pool.
<code>port</code>	<code>number</code> : The port number for this resource.
<code>propertySheet</code>	<code>reference</code> : <code>propertySheet</code>
<code>proxyCustomization</code>	<code>string</code> : This property displays the customization of the ElectricFlow proxy agent.

Object Type: resource

Description: A *resource* is associated with a server machine available to ElectricFlow for running steps. A resource has a logical name in addition to a host name. A resource can be associated with one or more pools. The resource may be a "proxy target" machine that communicates with the ElectricFlow server through an ElectricFlow agent proxy machine.

Property Name	Description
proxyHostName	name : The name of the ElectricFlow agent being used to proxy to another agent, the proxy target.
proxyPort	number : The port number of the ElectricFlow agent being used to proxy to another agent, the proxy target.
proxyProtocol	name : The name of the proxy agent protocol used for communication from the ElectricFlow proxy agent to the proxy target - default is SSH.
repositoryNames	A "new line" separated list of repository names.
resourceDisabled	boolean : A Boolean value indicating whether this resource was disabled.
resourceId	id : This is this resource's ID.
resourceName	name : The name of the resource or pool.
shell	name : The shell used to execute the step's commands on a resource. If no shell was defined on a step, the shell defined on the resource is used, or if no shell was defined on the resource, a default shell is used. For shells: The script name is inserted into the command at the position of a "{0}" marker in the command, or appended as a final argument if no marker is present. A filename suffix adjacent to the marker will be appended to the script name. For more information on shells, see the Shell definition in the Step Help topic.
stepCount	number : The current number of executing steps on this resource.
stepLimit	number : This property specifies the maximum number of steps that can run on this resource at one time.

Object Type: resource

Description: A *resource* is associated with a server machine available to ElectricFlow for running steps. A resource has a logical name in addition to a host name. A resource can be associated with one or more pools. The resource may be a "proxy target" machine that communicates with the ElectricFlow server through an ElectricFlow agent proxy machine.

Property Name	Description
trusted	<code>boolean</code> : If "true", this agent is trusted.
useSSL	<code>boolean</code> : A Boolean value indicating whether this resource uses SSL for communication. Transport Layer Security (TLS) has replaced Secure Sockets Layer version 3.0 (SSLv3) on the ElectricFlow web server and the ElectricFlow server.
workspaceName	<code>name</code> : The workspace name used by this resource.
zoneName	<code>string</code> : The name of the zone where this resource resides.

[\[back to top\]](#)

Object Type: resourcePool

Description: A *resource pool* is a container for a group of resources.

Property Name	Description
acl	<code>reference</code> : <code>acl</code>
autoDelete	<code>boolean</code> : If "true", the resource pool is deleted when the last resource is removed or deleted.
createTime	<code>date</code> : The time when this object was created.
description	<code>string</code> : A user-specified text description of the object.

Object Type: resourcePool

Description: A *resource pool* is a container for a group of resources.

Property Name	Description
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
lastResourceUsed	<code>name</code> : The name of the most recently used resource from the pool.
modifyTime	<code>date</code> : The time when this object was last modified.
orderingFilter	<code>string</code> : A Javascript block invoked when scheduling resources for a pool. Note: A Javascript block is not required unless you need to override the default resource ordering behavior.
owner	<code>name</code> : The person (user name) who created the object.
propertySheet	<code>reference</code> : propertySheet
resourcePoolDisabled	<code>boolean</code> : A Boolean value indicating whether this resource pool was disabled.
resourcePoolId	<code>id</code> : This resource pool's ID number.
resourcePoolName	<code>name</code> : The name of the resource pool.

[\[back to top\]](#)

Object Type: resourceUsage

Description: Provides usage information for all resources in the system.

For any job step running on a resource, there is a resource usage record that contains the ID and name of the job, job step, and resource.

Property Name	Description
jobId	<code>id</code> : This is this job's ID number, which is a UUID.

Object Type: resourceUsage

Description: Provides usage information for all resources in the system.

For any job step running on a resource, there is a resource usage record that contains the ID and name of the job, job step, and resource.

Property Name	Description
jobName	<code>name</code> : The name of the job.
jobStepId	<code>id</code> : The ID number of the job step.
jobStepName	<code>name</code> : The name of the job step.
licenseWaitTime	The time this job step had to wait because no license was found or available.
resourceId	<code>id</code> : The ElectricFlow-generated resource ID number.
resourceName	<code>name</code> : The name (or names) of the resource.
resourcePoolId	<code>id</code> : The ElectricFlow-generated resource pool's ID number.
resourcePoolName	<code>name</code> : The name of the resource pool.
resourceUsageId	<code>id</code> : The unique ID of the resource usage record.
resourceWaitTime	The time this job step had to wait because no resource was found or available. This could indicate that eligible resources for the step have reached their step limit, are in-use but the step requires a resource exclusively, or resources are down.
waitReason	Possible values are: <code>license</code> , <code>resource</code> , or <code>workspace</code> .
workspaceWaitTime	The time this job step had to wait because no workspace was found or available.

[\[back to top\]](#)

Object Type: <code>schedule</code> Description: <i>Schedules</i> are used to execute procedures and determine when specific procedures run.	
Property Name	Description
<code>acl</code>	<code>reference</code> : <code>acl</code>
<code>actualParameter</code>	<code>reference</code> : <code>propertySheet</code> An <i>actualParameter</i> is an object that provides the value for a parameter, which is passed to a procedure when it is invoked. Actual parameters can be provided for jobs and nested subprocedures within a job. Actual parameters are different from "formal parameters"- formal parameters define the parameters a procedure is expecting, and actual parameters provide values to use at run-time.
<code>beginDate</code>	<code>string</code> : The first day on which the schedule is active, in the form YYYY-MM-dd. For example, "2009-06-25"
<code>createTime</code>	<code>date</code> : The time when this object was created.
<code>credentialName</code>	<code>name</code> : The name of the <code>credential</code> being used for impersonation when the schedule launches a job.
<code>description</code>	<code>string</code> : A user-specified text description of the object.
<code>endDate</code>	<code>string</code> : The end of the range of dates the schedule is active, in the form: YYYY-MM-dd. The actual end date is not included in the range. For example, "2009-06-25"
<code>interval</code>	<code>string</code> : A floating point number that represents the time to wait between invocations of the schedule.
<code>intervalUnits</code>	<code>string</code> : These are the units to use when interpreting the <i>interval</i> value. The <i>units</i> can be <code>hours</code> , <code>minutes</code> , <code>seconds</code> , or <code>continuous</code> .
<code>lastModifiedBy</code>	<code>name</code> : This shows who (generally, a user name) last modified this object.
<code>lastRunTime</code>	<code>date</code> : The last time a job was launched by a schedule.

Object Type: <code>schedule</code> Description: <i>Schedules</i> are used to execute procedures and determine when specific procedures run.	
Property Name	Description
<code>misfirePolicy</code>	string: The policy the schedule uses when it is unable to launch a job at the scheduled time: <code>ignore</code> - wait until the next scheduled time to run <code>runOnce</code> - run immediately, then resume normal scheduling
<code>modifyTime</code>	date: The time when this object was last modified.
<code>monthDays</code>	string: This property specifies which days of the month this schedule should run - shown as a space-separated list of numbers (1-31).
<code>owner</code>	name: The person (user name) who created the object.
<code>priority</code>	string: Values can be <code>low</code> , <code>normal</code> (default), <code>high</code> , or <code>highest</code>
<code>procedureName</code>	name: The name of the <code>procedure</code> that contains this schedule.
<code>projectName</code>	name: The name of the <code>project</code> that contains this schedule.
<code>propertySheet</code>	reference: <code>propertySheet</code>
<code>scheduleDisabled</code>	boolean: A Boolean value indicating whether this schedule was disabled.
<code>scheduleId</code>	id: This schedule's ID number.
<code>scheduleName</code>	name: This property displays this schedule's name and differs from the <code>liveSchedule</code> property found under a job in that the <code>liveSchedule</code> property is written at job creation time only, and not changed, even if the schedule is renamed or deleted.
<code>startTime</code>	string: The time of day when this schedule should start running, in the form: <code>HH:mm:ss</code>

Object Type: schedule

Description: *Schedules* are used to execute procedures and determine when specific procedures run.

Property Name	Description
stopTime	string: The time of day when this schedule should stop running, in the form: HH:mm:ss
timeZone	string: A Java-compatible time zone string.
weekDays	string: This property specifies which days of the week this schedule should run. This is a space-separated list of MONDAY TUESDAY WEDNESDAY THURSDAY FRIDAY SATURDAY SUNDAY. (Note: These are not localized strings.)

[\[back to top\]](#)

Object Type: server

Description: This is the ElectricFlow server. There is only one such object in your database.

Read-only server properties

Two read-only server properties are available for use in a step, a script, or an email notifier, which can be used to identify the ElectricFlow server location.

`/server/hostname` - returns the ElectricFlow server name

`/server/hostIP` - returns the ElectricFlow server IP address

These properties are especially useful for building a URL link to an ElectricFlow web page. For example, a link in an email notifier that links back to a Job Details page.

A sample link: `https://$[/server/hostname]/commander/jobDetails.php?jobId=$[jobId]`

Property Name	Description
acl	reference: <code>acl</code>
createTime	date: The time when this object was created.
hostIP	string: The server's IP address.

Object Type: `server`

Description: This is the ElectricFlow server. There is only one such object in your database.

Read-only server properties

Two read-only server properties are available for use in a step, a script, or an email notifier, which can be used to identify the ElectricFlow server location.

`/server/hostname` - returns the ElectricFlow server name

`/server/hostIP` - returns the ElectricFlow server IP address

These properties are especially useful for building a URL link to an ElectricFlow web page. For example, a link in an email notifier that links back to a Job Details page.

A sample link: `https://$[/server/hostname]/commander/jobDetails.php?jobId=$[jobId]`

Property Name	Description
<code>hostname</code>	<code>name</code> : Generally refers to the computer name or IP address for the machine containing the ElectricFlow server.
<code>lastModifiedBy</code>	<code>name</code> : This shows who (generally, a user name) last modified this object.
<code>modifyTime</code>	<code>date</code> : The time when this object was last modified.
<code>owner</code>	<code>name</code> : The person (user name) who created the object.
<code>propertySheet</code>	<code>reference</code> : <code>propertySheet</code>
<code>serverId</code>	<code>id</code> : The server's ID number.
<code>serverName</code>	<code>string</code> : The ElectricFlow server's name - defaults to "server".

[\[back to top\]](#)

Object Type: state

Description: A *state* object belongs to a workflow and corresponds to the state definition, including a pointer to the workflow, formal parameters (cloned from definition), expanded actual parameters (from the last time a transition was taken to this state), and the "process" which includes the procedure or workflow (cloned from definition), unexpanded actual parameters (cloned from definition, expanded and passed to the process on entry), and the "last run" entity reference.

Property Name	Description
acl	<code>reference</code> : <code>acl</code>
active	<code>boolean</code> : If "true", the state of the workflow is active.
actualParameter	<code>reference</code> : <code>propertySheet</code> An <code>actualParameter</code> is an object that provides the value for a parameter, which is passed to a procedure when it is invoked. Actual parameters can be provided for jobs and nested subprocedures within a job. Actual parameters are different from "formal parameters"- formal parameters define the parameters a procedure is expecting, and actual parameters provide values to use at run-time.
createTime	<code>date</code> : The time when this object was created.
description	<code>string</code> : A user-specified text description of the object.
errorMessage	<code>string</code> : When the outcome is <code>error</code> , this property displays an error message description.
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.
projectName	<code>name</code> : The name of the <code>project</code> that contains this state.
propertySheet	<code>reference</code> : <code>propertySheet</code>
stateId	<code>id</code> : The ElectricFlow-generated ID number for the state.

Object Type: state

Description: A *state* object belongs to a workflow and corresponds to the state definition, including a pointer to the workflow, formal parameters (cloned from definition), expanded actual parameters (from the last time a transition was taken to this state), and the "process" which includes the procedure or workflow (cloned from definition), unexpanded actual parameters (cloned from definition, expanded and passed to the process on entry), and the "last run" entity reference.

Property Name	Description
stateName	name : The name of the state.
subjob	name : The name of the subjob.
subparameters	reference : propertySheet
subprocedure	name : The name of the nested procedure called when a step runs.
subproject	string : If a subprocedure argument was used, this is the name of the project where that subprocedure is found. By default, the current project is used.
substartingState	name : Name of the starting state for the workflow launched when the state is entered.
subworkflow	name : The name of the subworkflow - a workflow called by another workflow.
subworkflowDefinition	name : The name of the subworkflow definition.
workflow	name : The name of the workflow.

[\[back to top\]](#)

Object Type: stateDefinition

Description: A *stateDefinition* is a named object that belongs to a workflow definition, which includes specifications for whether or not the state is "startable", formal parameters, notifications, and the process. A process includes a procedure or workflow, the starting state (for workflows only), and actual parameters.

Property Name	Description
acl	<code>reference: acl</code>
actualParameter	<code>reference: propertySheet</code> An <i>actualParameter</i> is an object that provides the value for a parameter, which is passed to a procedure when it is invoked. Actual parameters can be provided for jobs and nested subprocedures within a job. Actual parameters are different from "formal parameters"- formal parameters define the parameters a procedure is expecting, and actual parameters provide values to use at run-time.
createTime	<code>date</code> : The time when this object was created.
description	<code>string</code> : A user-specified text description of the object.
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.
projectName	<code>name</code> : The name of the project that contains this state definition.
propertySheet	<code>reference: propertySheet</code>
startable	<code>boolean</code> : "True" means this state definition can be the initial state of an instantiated workflow.
stateDefinitionId	<code>id</code> : The ElectricFlow-generated ID number for the state definition.
stateDefinitionName	<code>name</code> : The name of the state definition.

Object Type: stateDefinition

Description: A *stateDefinition* is a named object that belongs to a workflow definition, which includes specifications for whether or not the state is "startable", formal parameters, notifications, and the process. A process includes a procedure or workflow, the starting state (for workflows only), and actual parameters.

Property Name	Description
subprocedure	<code>name</code> : The name of the nested procedure called when a step runs.
subproject	<code>string</code> : If a subprocedure argument was used, this is the name of the project where that subprocedure is found. By default, the current project is used.
substartingState	<code>name</code> : Name of the starting state for the workflow launched when the state is entered.
subworkflowDefinition	<code>name</code> : The name of the subworkflow definition.
workflowDefinitionName	<code>name</code> : The name of the workflow definition.

[\[back to top\]](#)

Object Type: step

Description: A *step* includes a command or script executed on a single resource and is the smallest unit of work ElectricFlow understands. Each step specifies a resource on which it should run (either the name of a specific machine or the name of a *pool* of equivalent machines, in which case ElectricFlow picks a machine from the pool). A step can be given a time limit and if the step does not complete within the specified time, ElectricFlow automatically aborts it.

Steps are ordered within a procedure and normally execute sequentially. However, it is possible to mark a consecutive range of steps for parallel execution, so all steps in that range run concurrently.

A step in one procedure can invoke another procedure instead of running a command. The invoking step provides parameters needed by the nested procedure.

Property Name	Description
<code>acl</code>	<code>reference</code> : <code>acl</code>
<code>actualParameter</code>	<code>reference</code> : <code>propertySheet</code> An <code>actualParameter</code> is an object that provides the value for a parameter, which is passed to a procedure when it is invoked. Actual parameters can be provided for jobs and nested subprocedures within a job. Actual parameters are different from "formal parameters"- formal parameters define the parameters a procedure is expecting, and actual parameters provide values to use at run-time.
<code>alwaysRun</code>	<code>boolean</code> : If true, this step runs even if the job is aborted before the step completes. Note that a force abort will abort an <code>alwaysRun</code> step.
<code>broadcast</code>	<code>boolean</code> : If true, this step will run on all resources in a pool.
<code>command</code>	<code>string</code> : This property specifies the command this step runs.
<code>condition</code>	<code>string</code> : If this element is not present, the event is ALWAYS triggered. If specified, the event is triggered only if the value of the condition argument is TRUE; if not true, a boolean value of "false" or "0" was used. Condition arguments can be a literal, a fixed value string, or a string subject to property expansion.
<code>createTime</code>	<code>date</code> : The time when this object was created.
<code>credentialName</code>	<code>name</code> : The credential name assigned to this step.

Object Type: step

Description: A *step* includes a command or script executed on a single resource and is the smallest unit of work ElectricFlow understands. Each step specifies a resource on which it should run (either the name of a specific machine or the name of a *pool* of equivalent machines, in which case ElectricFlow picks a machine from the pool). A step can be given a time limit and if the step does not complete within the specified time, ElectricFlow automatically aborts it.

Steps are ordered within a procedure and normally execute sequentially. However, it is possible to mark a consecutive range of steps for parallel execution, so all steps in that range run concurrently.

A step in one procedure can invoke another procedure instead of running a command. The invoking step provides parameters needed by the nested procedure.

Property Name	Description
description	<code>string</code> : A user-specified text description of the object.
errorHandling	<p>This is the error handling policy copied from the procedure step and indicates how the step responds to errors:</p> <p><code>failProcedure</code> - The current procedure continues, but the overall status is error (default).</p> <p><code>abortProcedure</code> - Aborts the current procedure, but allows already-running steps in the current procedure to complete.</p> <p><code>abortProcedureNow</code> - Aborts the current procedure and terminates running steps in the current procedure.</p> <p><code>abortJob</code> - Aborts the entire job, terminates running steps, but allows <code>alwaysRun</code> steps to run.</p> <p><code>abortJobNow</code> - Aborts the entire job and terminates all running steps, including <code>alwaysRun</code> steps.</p> <p><code>ignore</code> - Continues as if the step succeeded.</p>
exclusive	<code>boolean</code> : If true, this step acquires and retains its resource exclusively.
exclusiveMode	<code>string</code> : Possible values are: none, job, step, call
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.

Object Type: step

Description: A *step* includes a command or script executed on a single resource and is the smallest unit of work ElectricFlow understands. Each step specifies a resource on which it should run (either the name of a specific machine or the name of a *pool* of equivalent machines, in which case ElectricFlow picks a machine from the pool). A step can be given a time limit and if the step does not complete within the specified time, ElectricFlow automatically aborts it.

Steps are ordered within a procedure and normally execute sequentially. However, it is possible to mark a consecutive range of steps for parallel execution, so all steps in that range run concurrently.

A step in one procedure can invoke another procedure instead of running a command. The invoking step provides parameters needed by the nested procedure.

Property Name	Description
logFileName	<code>name</code> : The name of the log file produced by this step, relative to the job's workspace directory.
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.
parallel	<code>boolean</code> : If true, this step runs in parallel with other adjacent steps also marked to run in parallel.
postLogFileName	<code>name</code> : This property displays the log file name produced by this step's post processor.
postProcessor	<code>string</code> : This property displays the post processor name that is used to gather information about this step. Typically, this is either <code>postp</code> or <code>postp <options></code> , or an appropriate command or path including options to a postprocessor available on the ElectricFlow server.

Object Type: step

Description: A *step* includes a command or script executed on a single resource and is the smallest unit of work ElectricFlow understands. Each step specifies a resource on which it should run (either the name of a specific machine or the name of a *pool* of equivalent machines, in which case ElectricFlow picks a machine from the pool). A step can be given a time limit and if the step does not complete within the specified time, ElectricFlow automatically aborts it.

Steps are ordered within a procedure and normally execute sequentially. However, it is possible to mark a consecutive range of steps for parallel execution, so all steps in that range run concurrently.

A step in one procedure can invoke another procedure instead of running a command. The invoking step provides parameters needed by the nested procedure.

Property Name	Description
precondition	<p>string: By default, if a step has no precondition, it will run when scheduled. Set this property to make a step wait until one or more dependent conditions are met. When a job step is eligible to transition from pending to runnable, a <i>precondition</i> is evaluated.</p> <p>A <i>precondition</i> is a fixed text or text embedding property reference that is evaluated to TRUE or FALSE. An empty string, a \"0\" or \"false\" is interpreted as FALSE. Any other result string is interpreted as TRUE. The step will block until the precondition is TRUE.</p>
procedureName	name: The name of the procedure that contains this <i>step</i> .
projectName	name: The name of the project that contains this step.
propertySheet	reference: <code>propertySheet</code>
releaseExclusive	boolean: A Boolean value indicating whether this step should release its resource upon completion.
releaseMode	string: Possible values are: none, release, releaseToJob
resourceName	name: The resource's name this step should use to run on.

Object Type: step

Description: A *step* includes a command or script executed on a single resource and is the smallest unit of work ElectricFlow understands. Each step specifies a resource on which it should run (either the name of a specific machine or the name of a *pool* of equivalent machines, in which case ElectricFlow picks a machine from the pool). A step can be given a time limit and if the step does not complete within the specified time, ElectricFlow automatically aborts it.

Steps are ordered within a procedure and normally execute sequentially. However, it is possible to mark a consecutive range of steps for parallel execution, so all steps in that range run concurrently.

A step in one procedure can invoke another procedure instead of running a command. The invoking step provides parameters needed by the nested procedure.

Property Name	Description
shell	<code>name</code> : The shell used to execute the step's commands on a resource. The script name is inserted into the command at the position of a "{0}" marker in the command, or appended as a final argument if no marker is present. A filename suffix adjacent to the marker will be appended to the script name. For more information on shells, see the Shell definition in the Step Help topic.
stepId	<code>id</code> : The step's ID number.
stepName	<code>name</code> : The step's name.
subprocedure	<code>name</code> : The nested procedure name to call when this step executes.
subproject	<code>name</code> : This property is displayed if the subprocedure element is present and specifies the project to which the subprocedure belongs (by default, the current project is used.)
timeLimit	<code>number</code> : A floating point number that specifies the maximum length of time this step is allowed to run.
timeLimitUnits	The units to use when interpreting the time limit. Units can be <code>hours</code> , <code>minutes</code> , or <code>seconds</code> .
workingDirectory	<code>name</code> : The name of the step's working directory.
workspaceName	<code>name</code> : The workspace name used by this step.

[\[back to top\]](#)

Object Type: transition

Description: A *transition* object belongs to a state and corresponds to the transition definition, and includes the target state (unexpanded actual parameters cloned from definition, expanded and passed to the target state on entry), the Javascript condition, and the trigger (`onEnter|onStart|onCompletion|manual`).

Property Name	Description
<code>acl</code>	<code>reference</code> : <code>acl</code>
<code>actualParameter</code>	<code>reference</code> : <code>propertySheet</code> An <i>actualParameter</i> is an object that provides the value for a parameter that is passed to the target state when the transition is taken.
<code>condition</code>	<code>string</code> : If empty or non-zero, the transition is allowed to trigger. If set to "0", the transition is ignored.
<code>createTime</code>	<code>date</code> : The time when this object was created.
<code>description</code>	<code>string</code> : A user-specified text description of the object.
<code>index</code>	The numeric index of a transition that indicates the transition order in the state definition's transition list.
<code>lastModifiedBy</code>	<code>name</code> : This shows who (generally, a user name) last modified this object.
<code>modifyTime</code>	<code>date</code> : The time when this object was last modified.
<code>owner</code>	<code>name</code> : The person (user name) who created the object.
<code>projectName</code>	<code>name</code> : The name of the <code>project</code> that contains this transition.
<code>propertySheet</code>	<code>reference</code> : <code>propertySheet</code>
<code>stateName</code>	<code>name</code> : The name of the state.
<code>targetState</code>	<code>string</code> : The target state for the transition definition.
<code>transitionId</code>	<code>id</code> : The ElectricFlow-generated ID for the transition.

Object Type: transition

Description: A *transition* object belongs to a state and corresponds to the transition definition, and includes the target state (unexpanded actual parameters cloned from definition, expanded and passed to the target state on entry), the Javascript condition, and the trigger (onEnter|onStart|onCompletion|manual).

Property Name	Description
transitionName	<code>name</code> : The name of the transition.
trigger	<code>string</code> : Possible values are: onEnter - before any actions onStart - after a subjob or workflow is created onCompletion - after a subjob or workflow completes manual - when a user manually requests a transition
workflowName	<code>name</code> : The name of the workflow.

[\[back to top\]](#)**Object Type: transitionDefinition**

Description: A *transitionDefinition* is a named object that belongs to a state definition, which includes the target state definition (with actual parameters to the target state, the Javascript condition, and the trigger (onEnter|onStart|onCompletion|manual).

Property Name	Description
acl	<code>reference</code> : acl
actualParameter	<code>reference</code> : <code>propertySheet</code> An <i>actualParameter</i> is an object that provides the value for a parameter passed to the target state.
condition	<code>string</code> : If empty or non-zero, the transition is allowed to trigger. If set to "0", the transition is ignored.
createTime	<code>date</code> : The time when this object was created.

Object Type: transitionDefinition

Description: A *transitionDefinition* is a named object that belongs to a state definition, which includes the target state definition (with actual parameters to the target state, the Javascript condition, and the trigger (onEnter|onStart|onCompletion|manual)).

Property Name	Description
description	<code>string</code> : A user-specified text description of the object.
index	The numeric index of a transition that indicates the transition order in the state definition's transition list.
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.
projectName	<code>name</code> : The name of the project that contains this transition definition.
propertySheet	<code>reference</code> : propertySheet
stateDefinitionName	<code>name</code> : The name of the state definition.
targetState	<code>string</code> : The target state for the transition definition.
transitionDefinitionId	<code>id</code> : The ElectricFlow-generated ID for the transition definition.
transitionDefinitionName	<code>name</code> : The name of the transition definition.
trigger	<code>string</code> : Possible values are: onEnter – before any actions onStart – after a subjob or workflow is created onCompletion – after a subjob or workflow completes manual – when a user manually requests a transition
workflowDefinitionName	<code>name</code> : The name of the workflow definition.

[\[back to top\]](#)

Object Type: `User`

Description: *User* refers to any user currently logged into the ElectricFlow system or any user who may use ElectricFlow, but may not be currently logged in.

Property Name	Description
<code>acl</code>	<code>reference:</code> <code>acl</code>
<code>createTime</code>	<code>date:</code> The time when this object was created.
<code>email</code>	<code>name:</code> Displays this user's email address.
<code>fullUserName</code>	<code>name:</code> The user's real name.
<code>lastModifiedBy</code>	<code>name:</code> This shows who (generally, a user name) last modified this object.
<code>modifyTime</code>	<code>date:</code> The time when this object was last modified.
<code>mutable</code>	<code>boolean:</code> If true, the user is editable within ElectricFlow via the web UI or the <code>modifyUser</code> API.
<code>owner</code>	<code>name:</code> The person (user name) who created the object.
<code>propertySheet</code>	<code>reference:</code> <code>propertySheet</code>
<code>providerName</code>	<code>name:</code> The name of the <code>directory provider</code> that controls this user.
<code>userId</code>	<code>id:</code> A unique ID number for a user object.
<code>userName</code>	<code>name:</code> Displays this user's name and also appears in a group object and displays the name of a user who belongs to this group.

[\[back to top\]](#)

Object Type: workflow

Description: A *workflow* object includes a pointer to the workflow definition, sets of states, the active state, the starting state, parameters to the initial state, "complete" - a boolean value determining whether or not the workflow is complete, and a log containing the transactions taken and user events.

Property Name	Description
acl	<code>reference</code> : acl
activeState	<code>string</code> : The name of the active state on the workflow object.
actualParameter	<code>reference</code> : <code>propertySheet</code> An <i>actualParameter</i> is an object that provides the value for a parameter that is passed to the target state when the transition is taken.
callingState	<code>string</code> : The full property path to the state that created this workflow.
callingStateId	<code>id</code> : The ID number for the full property path to the state that created this workflow.
completed	<code>boolean</code> : If "true", the workflow is completed and no additional transactions will be evaluated.
createTime	<code>date</code> : The time when this object was created.
deleted	<code>boolean</code> : The object was marked for background deletion. Possible values are "0" or "1". Default is "0" (not set).
elapsedTime	The time this object ran from start to finish.
finish	<code>date</code> : The date/time when this object finished.
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
launchedByUser	<code>string</code> : The name of the user or project principal that explicitly launched the workflow.
liveWorkflowDefinition	<code>name</code> : The current workflow definition name for the workflow definition from which the workflow was created.

Object Type: workflow

Description: A *workflow* object includes a pointer to the workflow definition, sets of states, the active state, the starting state, parameters to the initial state, "complete" - a boolean value determining whether or not the workflow is complete, and a log containing the transactions taken and user events.

Property Name	Description
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.
projectName	<code>name</code> : The name of the project that contains this workflow.
propertySheet	<code>reference</code> : propertySheet
start	<code>date</code> : The date/time when this workflow began to run.
startingState	<code>string</code> : The initial state of the workflow.
workflowDefinitionName	<code>name</code> : The name of the workflow definition.
workflowId	<code>id</code> : The ElectricFlow-generated ID for the workflow.
workflowName	<code>name</code> : The name of the workflow.

[\[back to top\]](#)**Object Type: workflowDefinition**

Description: A *workflowDefinition* object contains state and transition definitions, including the workflow name template (analogous to a job name template on a procedure), and ordered sets of state and transition definitions.

Property Name	Description
acl	<code>reference</code> : acl
createTime	<code>date</code> : The time when this object was created.

Object Type: workflowDefinition

Description: A *workflowDefinition* object contains state and transition definitions, including the workflow name template (analogous to a job name template on a procedure), and ordered sets of state and transition definitions.

Property Name	Description
description	<code>string</code> : A user-specified text description of the object.
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.
projectName	<code>name</code> : The name of the <code>project</code> that contains this workflow definition.
propertySheet	<code>reference</code> : <code>propertySheet</code>
workflowDefinitionId	<code>id</code> : The ElectricFlow-generated ID for the workflow definition.
workflowDefinitionName	<code>name</code> : The name of the workflow definition.
workflowNameTemplate	<code>string</code> : Template used to determine the default names for workflows launched from a workflow definition.

[\[back to top\]](#)

Object Type: workspace

Description: ElectricFlow provides each job step with an area on the disk it can use for "working files" and results. This disk area is called a job *workspace*. (The job workspace defaults to a directory under the workspace directory.) A job step can create whatever files it needs within its workspace, and ElectricFlow automatically places files such as step logs in the workspace.

Normally, a single workspace is shared by all steps in a job, but different steps within a job could use different workspaces. The location of the job step workspace is displayed on the Job Details page for the job under "Details" for the step.

Property Name	Description
acl	<code>reference:</code> acl
agentDrivePath	<code>name:</code> The drive-letter based mount point for this workspace on Windows agents.
agentUncPath	<code>name:</code> The UNC path for this workspace on Windows agents.
agentUnixPath	<code>name:</code> The UNIX path for this workspace on UNIX agents.
createTime	<code>date:</code> The time when this object was created.
credentialName	<code>name:</code> The name of the credential being used when a resource connects to a network share while setting up the workspace.
description	<code>string:</code> A user-specified text description of the object.
lastModifiedBy	<code>name:</code> This shows who (generally, a user name) last modified this object.
local	<code>boolean:</code> If "true", this workspace is local.
modifyTime	<code>date:</code> The time when this object was last modified.
owner	<code>name:</code> The person (user name) who created the object.
propertySheet	<code>reference:</code> propertySheet

Object Type: workspace

Description: ElectricFlow provides each job step with an area on the disk it can use for "working files" and results. This disk area is called a job *workspace*. (The job workspace defaults to a directory under the workspace directory.) A job step can create whatever files it needs within its workspace, and ElectricFlow automatically places files such as step logs in the workspace.

Normally, a single workspace is shared by all steps in a job, but different steps within a job could use different workspaces. The location of the job step workspace is displayed on the Job Details page for the job under "Details" for the step.

Property Name	Description
<code>workspaceDisabled</code>	<code>boolean</code> : A Boolean value that indicates whether this workspace was disabled.
<code>workspaceId</code>	<code>id</code> : The workspace's ID number.
<code>workspaceName</code>	<code>name</code> : The workspace's name.
<code>zoneName</code>	<code>name</code> : The name of the zone where this workspace resides.

[\[back to top\]](#)

Object Type: Zone

Description: ElectricFlow provides the ability to create zones—often used to enhance security.

- A zone is a way to partition a collection of agents to secure them from use by other groups. For example, you might choose to create a developers zone, a production zone, and a test zone—agents in one zone cannot directly communicate with agents in another zone.
- A *default* zone is created during ElectricFlow installation. The server implicitly belongs to the default zone, which means all agents in this zone can communicate with the server directly (without the use of a gateway).
- Each zone can have one or more "gateway agents", which you define. Gateway agents are used for communication from one zone to another zone. For more information, see the [Gateways](#) Help topic.

Property Name	Description
<code>acl</code>	<code>reference</code> : <code>acl</code>

Object Type: Zone

Description: ElectricFlow provides the ability to create zones—often used to enhance security.

- A zone is a way to partition a collection of agents to secure them from use by other groups. For example, you might choose to create a developers zone, a production zone, and a test zone—agents in one zone cannot directly communicate with agents in another zone.
- A *default* zone is created during ElectricFlow installation. The server implicitly belongs to the default zone, which means all agents in this zone can communicate with the server directly (without the use of a gateway).
- Each zone can have one or more "gateway agents", which you define. Gateway agents are used for communication from one zone to another zone. For more information, see the [Gateways](#) Help topic.

Property Name	Description
createTime	date : The time when this object was created.
description	string : A user-specified text description for the object.
lastModifiedBy	name : This person (generally, a user name) last modified this object.
modifyTime	date : The time when this object was last modified.
owner	name : The person (user name) who created the object.
propertySheet	reference : propertySheet
resources	string : A space-separated list of resources contained in this zone.
zoneId	id : The ElectricFlow-generated ID for this zone.
zoneName	string : The name of the zone.

Property error codes

The following list provides error codes that may appear as a value within an `errorCode` property.

ABORTED

ACCESS_DENIED

AGENT_BAD_WORKINGDIR

AGENT_FAILED_CONNECT_BROKER

AGENT_FAILED_CREATE_FILE
AGENT_FAILED_CREATE_WORKSPACE
AGENT_FAILED_IMPERSONATION
AGENT_FAILED_MAP_DRIVE
AGENT_FAILED_PROXYTARGET_PING
AGENT_INCOMPATIBLE_VERSION
AGENT_INTERNAL_ERROR
AGENT_INVALID_CWD
AGENT_INVALID_MESSAGE
AGENT_INVALID_PING_TOKEN
AGENT_INVALID_WORKSPACE
AGENT_IO_CONNECTION_REFUSED
AGENT_IO_CONNECTION_RESET
AGENT_IO_ERROR
AGENT_IO_NO_ROUTE_TO_HOST
AGENT_IO_PORT_UNREACHABLE
AGENT_MALFORMED_XML
AGENT_NONABSOLUTE_PATH
AGENT_NONEXISTENT_DIR
AGENT_NONEXISTENT_FILE
AGENT_PING_FAILED
AGENT_STREAM_STOPPED
AGENT_TIMEOUT
AGENT_UNKNOWN_CMDID
AGENT_UNKNOWN_COMMAND
AGENT_WRONG_FILE_TYPE
BAD_AGENT_RESPONSE
BAD_LOGFILE_PATH
CANCELED
CIRCULAR_PROCEDURE_REFERENCE
CORRUPT_CREDENTIAL

DUPLICATE_JOB_NAME
EMPTY_SUBPROCEDURE
FAILED_JOB_RENAME
FORMAL_PARAMETER_ERROR
INTERNAL_ERROR
INVALID_EMAIL_HEADER
MY_EVENT_EXPANSION_ERROR
NO_EMAIL_HEADERS_SEPARATOR
NONEXISTENT_EMAIL_CONFIG
NONEXISTENT_PROCEDURE
NONEXISTENT_RESOURCE
NONEXISTENT_WORKSPACE
NOTIFIER_EXPANSION_ERROR
POST_PROCESSOR_ERROR
POST_PROCESSOR_OUTPUT_FAILURE
PROPERTY_REFERENCE_ERROR
RESOURCE_WITHOUT_HOSTNAME
SERVER_SHUTDOWN
TIMEOUT
UNKNOWN_HOST

Property - create new or edit existing property

Use this pop-up window to create or modify a custom property attached to an object.

Enter information in the fields as follows:

Field Name	Description
Name	<p>The property name can be an arbitrary text string, but unless you are an experienced ElectricFlow user avoid using slashes and brackets as a part of the property name.</p> <p>Note: The name "properties" is NOT a valid property name.</p>

Field Name	Description
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code>
Value	This value can be an arbitrary text string.
Expandable	"uncheck" this box if you do not want to allow property expansion.

Click **OK** to save your new or modified property information.

For more information on creating and using properties, see the [Properties](#) Help topic.

Nested Property Sheet

Use this pop-up window to define or modify a nested property sheet.

1. Enter information in the fields as follows:

Name—The property name can be an arbitrary text string.

Description—A plain text or HTML description for this object. If using HTML, you must surround your text with `<html> ... </html>` tags. The only HTML tags allowed in the text are: `<a>` `` `
` `<div>` `<dl>` `` `<i>` `` `` `<p>` `<pre>` `` `<style>` `<table>` `<tc>` `<td>` `<th>` `<tr>` ``

2. Click **OK** to save your new or modified nested property sheet information.

For more information on creating and using properties, see the [Properties](#) Help topic.

Reports

If you have upgraded ElectricFlow this page displays historical reports (Cross Project Summary, Daily Summary, and Resource Summary) from your previous ElectricFlow version. These reports are no longer being updated.

Click the Historic/web server local reports link to see any previously created custom reports.

- Cross Project Summary - this is a 30-day report summary
- Daily Summary - this is a daily report showing all the procedures that ran
- Resource Summary - this report shows resource usage for a specific day
- Historic/web server local reports

For more information, see the [Reports](#) Help topic.

Create a new report

Use this page to create a new report. Select one of the report-type tabs to begin.

Multiple Series Report

Field and pull-down menu descriptions:

Field/Menu Name	Description
Report Title	<p>This is your report title. Type over the default report name, choosing any unique name for your report.</p> <div> IMPORTANT: Do not use special characters in the report title. Some examples of special characters are : / ? # [] @ ! , \$ & ' () * + , ; = </div>
Saved Filter	<p>Project - Click your mouse inside this field to see a list of projects from which to make a selection.</p> <p>Filter - After selecting a project, this drop-down menu will contain any available filters for this project.</p>
Time Period	Use the drop-down menu to select the time period.
Create thumbnail?	Check this box if you would like to make a thumbnail view of this report available on your Home page.
Object Type	Use the drop-down menu to select an object.
Table column choices	<p>Chart Type - Use the drop-down menu to choose the chart type.</p> <p>Function - Use the drop-down menu to choose the function you need.</p> <p>Property Name - Use the default property value or delete the text and click your mouse in the blank field to see a list of possible properties.</p> <p>Display Name - You can choose a different unique Display Name.</p> <p>Stacked - Select this checkbox to see your report results "stacked" versus overlaid.</p> <p>The "X" icon - Click this icon to delete any row you no longer need.</p>

Field/Menu Name	Description
Add Series button	Select the Add Series button if you would like to create additional table entries for additional report information.
Chart Options	Use the down-arrow to adjust the Time Grouping and see the defaults for the X and Y axis.
Table Options	Use the down-arrow to adjust the Time Grouping and see the default for the Time Column label.
Advanced	Use the down-arrow to select a Locale or add a Credential. Delete the Resource if you choose, then click your mouse in the blank field to see a list of other resources you might prefer to use.
Run Report button	Use this button to go to the Job Details page to run the report.
Create Schedule button	This button displays a box with a default schedule name that you can change if you choose. Click OK to go to the Job Details page.
Cancel button	Use this bottom to cancel the create report operation.

Category Report

Field and pull-down menu descriptions:

Field/Menu Names	Description
Report Title	This is your report title. Type over the default report name, choosing any unique name for your report. IMPORTANT: Do not use special characters in the report title. Some examples of special characters are : / ? # [] @ ! , \$ & ' () * + , ; =
Saved Filter	Project - Click your mouse inside this field to see a list of projects from which to make a selection. Filter - After selecting a project, this drop-down menu will contain any available filters for this project.
Time Period	Use the drop-down menu to select the time period.

Field/Menu Names	Description
Create thumbnail?	Check this box if you would like to make a thumbnail view of this report available on your Home page.
Object Type	Use the drop-down menu to select an object.
Object Property	You can delete the default value and click your mouse inside the blank field to see a list of properties from which to choose.
Advanced	Use the down-arrow to select a Locale or add a Credential. Delete the Resource if you choose, then click your mouse in the blank field to see a list of other resources you might prefer to use.
Run Report button	Use this button to go to the Job Details page to run the report.
Create Schedule button	This button displays box with a default schedule name that you can change if you choose. Click OK to go to the Job Details page.
Cancel button	Use this bottom to cancel the create report operation.

Count Over Time Report

Field and pull-down menu descriptions:

Field/Menu Name	Description
Report Title	<p>This is your report title. Type over the default report name, choosing any unique name for your report.</p> <div style="border: 1px solid black; padding: 5px; background-color: #e6f2ff;"> <p>IMPORTANT: Do not use special characters in the report title. Some examples of special characters are : / ? # [] @ ! , \$ & ' () * + , ; =</p> </div>
Saved Filter	<p>Project - Click your mouse inside this field to see a list of projects from which to make a selection.</p> <p>Filter - After selecting a project, this drop-down menu will contain any available filters for this project.</p>
Time Period	Use the drop-down menu to select the time period.

Field/Menu Name	Description
Create thumbnail?	Check this box if you would like to make a thumbnail view of this report available on your Home page.
Object Type	Use the drop-down menu to select an object.
Chart Options	Use the down-arrow to adjust the Time Grouping and see the defaults for the X and Y axis.
Table Options	Use the down-arrow to adjust the Time Grouping and see the default for the Time Column label.
Advanced	Use the down-arrow to select a Locale or add a Credential. Delete the Resource if you choose, then click your mouse in the blank field to see a list of other resources you might prefer to use.
Run Report button	Use this button to go to the Job Details page to run the report.
Create Schedule button	This button displays box with a default schedule name that you can change if you choose. Click OK to go to the Job Details page.
Cancel button	Use this bottom to cancel the create report operation.

Procedure Usage Report

Field and pull-down menu descriptions:

Field/Menu Name	Description
Report Title	<p>This is your report title. Type over the default report name, choosing any unique name for your report.</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p>IMPORTANT: Do not use special characters in the report title. Some examples of special characters are : / ? # [] @ ! , \$ & ' () * + , ; =</p> </div>

Field/Menu Name	Description
Project	<p>Click your mouse inside this blank field to see a list of projects. This does not have to be the current project.</p> <p>You can generate reports that show when one project's procedures call procedures in another project. You can also use the '%' wildcard, for example,</p> <p>Project: %</p> <p>or</p> <p>Project: D%</p>
Procedure pattern	Enter a SQL string match pattern for procedures to include in the report. The default % will use all procedures.
Advanced	Use the down-arrow to select a Locale or add a Credential. Delete the Resource if you choose, then click your mouse in the blank field to see a list of other resources you might prefer to use.
Run Report button	Use this button to go to the Job Details page to run the report.
Create Schedule button	This button displays box with a default schedule name that you can change if you choose. Click OK to go to the Job Details page.
Cancel button	Use this bottom to cancel the create report operation.

Resources

The **Cloud > Resources** page displays all resources known to this ElectricFlow server and provides easy access to resource configuration and all other resource management functionality.

- Each resource has a logical name—a unique name to distinguish this resource from other resource names.
- Each resource refers to an agent machine by its host name.
- Each resource can be assigned to one or more pools.

A *pool* is a group of interchangeable resources. For example, you may have a pool of Windows servers.

If you name a pool in a procedure step, ElectricFlow can assign any resource in the pool to that step, which allows ElectricFlow to choose a lightly loaded resource. You can change the resources in a pool without having to modify any procedures that use the pool.

- Each resource can be assigned to a zone. A default zone is created during ElectricFlow installation and all resources, by default, are members of that zone until they are assigned to a different zone.

A *zone* is a collection of agents. Every agent, and all resources defined on that agent, belong to only one zone. When ElectricFlow is installed, a *default* zone is created. If a zone is deleted, all agents in that zone are moved to the *default* zone. The ElectricFlow server resides in the default zone.

Note: The *default* zone **cannot** be deleted.

- Several resources can correspond to the same physical host or agent machine.
- When you specify a resource name in a procedure step, the step executes on that resource.

Supported Resource Types

- **Standard**—This type specifies a machine running the ElectricFlow agent on one of the supported agent platforms, as specified in the *ElectricFlow Installation Guide*.
- **Proxy**—This type requires SSH keys for authentication. You can create proxy resources (agents and targets) for ElectricFlow to use on other remote platforms or hosts that exist in your environment.
 - Proxy *agent*—This is an agent on a supported Linux or Windows platform, used to proxy commands to an otherwise unsupported platform. A proxy agent is an ElectricFlow agent, channeling to a proxy target. Proxy agents have limitations, such as the inability to work with plugins or communicate with ectool commands.
 - Proxy *target*—This is a machine on an unsupported platform that can run commands through an SSH server.

Help topic "Best Practice": We recommend reviewing this Help topic in its entirety to become familiar with new resource management features and expanded functionality. Later, when using the Resources page, use the following quick links to go to the Help section that you need to review.

[Icons, Links, and Buttons Above the Table](#)

[Table view - column descriptions](#)

[Grid view](#)

[Filters pane](#)

[New Resource panel](#)

[New Proxy Resource panel](#)

[Edit Resource panel](#)

[Edit Proxy Resource panel](#)

[Resource Details panel](#)

[Switching a non-trusted agent to trusted](#)

[Install or Upgrade Remote Agents](#)

Resource Page Information and Functions

Immediately after the *Resources* title above the table, you can see how many licensed resources are currently in use.




Links, Icons, and Buttons Above the Table

Hover your mouse over an icon or button to see the available actions. You can select one of these actions on a single resource or on multiple resources (selected using the check box in the first column) to perform the action on one or more resources at the same time.

- Select one or more resources from the table, and then select the button to perform the task or action.



- **Exception:** When you click the **Action** icon and select **Create Resource** or **Create Proxy Resource**, a panel opens where you can create one of these new objects. You can create only one object at a time.

Icon or Link Name	Description
	<p>Select one of these actions:</p> <ul style="list-style-type: none"> • Create Resource—The New Resource panel opens. For information about the fields, see the New Resource section. • Create Proxy Resource—The New Proxy Resource panel opens. For information about the fields, see the New Proxy Resource section. • Install Resource(s)—A dialog box opens where you enter information about the agents to install. For more information, see the Install/Upgrade Remote Agents section. <p>Note: If you are upgrading one or more agents or resources, the label on this option is Upgrade Resource(s), which is available only for Linux agents . You cannot install or upgrade Windows agents using this process.</p> <div style="border: 1px solid black; padding: 5px; background-color: #e6f2ff;"> <p>IMPORTANT: If the EC-AgentManagement plugin is not installed, you will not see this option.</p> </div>
	Copy Resource —Makes a duplicate copy of one or more resources that you selected.
	<p>Delete Resource—Deletes one or more resources that you selected.</p> <p>Note: If a resource is on a "gateway agent", the gateway will be deleted.</p>










Icon or Link Name	Description
	<p>Ping Resource—ElectricFlow pings one or more resources to check if they are available.</p> <p>When ElectricFlow "pings" a resource, it sends a message to the agent (for the resource) to make sure the agent is alive and running a version of software compatible with the ElectricFlow server. After the ping completes (or fails), the page refreshes to reflect the current resource state.</p> <p>Note: A gateway must exist before you ping a resource in a remote zone.</p>
	<p>Enable Resource—Enables one or more resources that you selected in the check boxes in the first column..</p>
	<p>Disable Resource—Disables one or more resources that you selected in the check boxes in the first column.</p>
	<p>Select one of these actions:</p> <ul style="list-style-type: none"> • When you click the button with no resources selected, the <i>Select some resources to edit</i> message appears. • When you select resources and click this button, these options appear: <ul style="list-style-type: none"> • Add to Pool—Adds the selected resources to a pool. • Remove from Pool—This is available only if all the selected resources have at least one pool in common. If not, this link becomes Remove from Pool <name>. • Move to Zone—Moves the selected resources to a zone. • Set Trusted—This is not available if the selected resources are already <i>trusted</i>. If not, all the selected resource will be changed to <i>trusted</i>. • Unset Trusted—This is available only if all the selected resources are trusted. If not, the selected resources will be changed to <i>untrusted</i>.
 or	<p>Select this button to add this Resources page to your Home page for one-click access to return to this page. If the icon is yellow, the page is already accessible from your Home page .</p>
New Search link	<p>Use this link to go to the Define Search page if you need to search for a particular object or set of objects. For example, job, resource, artifact, workflow, and so on. Use the "back" button to return to the Resources page.</p>






Table view - column descriptions

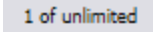
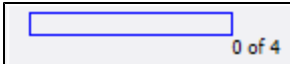

Not all resource configuration information is displayed within the table, but you can see all other information about the resource when you edit the resource or access other slide-out panels. For example, the action to see or add access control privileges to a resource is available on the Edit Resource panel or from the **Access Control** link on that panel.

Note: Most column headings are "sortable". Click on a column heading to re-sort the information in that column.

Column name / icon	Description
	Select this check box to enable the corresponding resource in that row for a single or batch action. For example, deleting or enabling multiple resources at the same time is a batch action. Note: If you select this icon in the table header row, it will select or deselect all resources in the table.
	Status of the agent. If an agent machine is not available, any resources associated with that host are not be available. Note: If an agent is down, the ElectricFlow server is not able to provide additional information about the cause.

Column name / icon	Description
Name	<p>Name of the resource. The icon to the left of the resource name indicates whether this resource is <i>enabled</i> or <i>disabled</i>. These icons are: Enabled  or Disabled </p> <p>If a resource is disabled, no job steps are assigned to it. If a step requests a particular resource and that resource is disabled, the step waits until the resource becomes enabled again. If the resource is in a pool, ElectricFlow uses any resource in the pool to satisfy the request for the resource. Other actions:</p> <ul style="list-style-type: none"> To change the enabled or disabled icon, select the check box (in first column) in the row for one or more resources, and then select the enabled or disabled icon above the table. Select multiple resources if this is a batch change. <p>Note: You can change the resource enable/disable specification from the Edit Resource panel too, but this is an individual operation for one particular resource.</p> <ul style="list-style-type: none"> Click on a resource name to see the Resource Details panel - this panel displays how your resources is configured currently. To make changes, click the Edit link to go to the Edit Resource panel.
Pools	<p>A space-separated list of pool names where this resource is a member.</p> <p>Note: Spaces are not allowed within a pool name.</p>
Type	<p>A resource can be <i>static</i> or <i>dynamic</i>.</p> <ul style="list-style-type: none"> A static resource is part of your system or network (not in the cloud), such as a server, database, or agent machine. A dynamic resource is a cloud resource that can be provisioned and later spun up on-demand when an application is deployed.
Resource Template Name	<p>The name of the resource template to which the resource is applied.</p> <p>Note: Only dynamic resources can be applied to a resource template.</p>
Time running	<p>If the resource is a dynamic resource, how long it has been running.</p>

Column name / icon	Description
Job Status	<p>Indicates the current job status with the following icons:</p> <p> <i>Running job</i> - a job is still running, which means the resource is currently being used for one or more jobs. The names of all jobs using the resource are displayed next to the icon.</p> <p> <i>Success</i> - the job running on this resource completed successfully.</p> <p> <i>Error</i> - the job running on this resource encountered an error and may or may not have completed. One or more error messages will be listed in this column.</p> <p> <i>Warning</i> - the job running on this resource encountered a warning and may or may not have completed. One or more warning messages will be listed in this column.</p> <p> <i>Busy</i> - ElectricFlow is attempting to refresh the job status, but the resource is not responding or the UI cannot parse the response. Changing the value of 'Agent Host Name' to the IP address can sometimes resolve this issue.</p>
Description	<p>A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code>.</p>
Zone	The name of the zone where this resource is a member.
HTTPS & Host	The "Connection Type" you specified when this resource was created.

Column name / icon	Description
Step Load	<p>Resource <i>load</i> is indicated by one of the following:</p> <p></p> <p>- this notation indicates 1 step is using this resource and an unlimited number of steps can use this resource at the same time. If 5 steps are using this resource, you would see "5 of unlimited".</p> <p></p> <p>- this notation indicates that no steps are running on this resource and the step limit is set to 4. The bar indicates graphically how close you are to the step limit.</p> <p>Note: Use the Create/Edit Resource panels to define the number of steps that can run on a resource. Setting the Step Limit to "0" or leaving the field blank defaults to NO step limit.</p>
Actions	<p>Actions you can perform on the resource, including:</p> <p>Copy—Create a new resource by copying the this resource.</p> <p>Delete—Delete this resource.</p> <p>Tear Down—Remove the dynamic resource from a dynamic environment.</p> <p></p> <p>Track Changes --Open the Change History for the resource.</p>

Grid view



Two icons above the Filters pane:

- The first icon (on the left) is the default *Table view* icon.
- The second icon (on the right) displays the resource *Grid view*.

Use the drop-down menus to filter the Grid view:

- Grouping - Select the resource "grouping" you want to see—choose Pool, Job, Operating System, Platform, Proxy Agent, Version, or Zone.
For example, if you have multiple resource pools defined and chose Pool, you will see groups labeled with your various pool names and a group (No Pool) for resources not included in a pool.

- Coloring - Select what you want to view in each group—choose Platform, Step Count/Limit, Status, Held Exclusive, Version, or Operating System to sort your group.
Note: Colors are dynamically assigned per the view you select. Within a single view, the same color "square" implies similar information.
- Hovering your mouse over a grid "square" provides more detailed information about that resource.

Filters pane

Use the Filters pane to filter resources shown in either the Table or Grid view. The fields in the Filter pane correspond to columns in the Table view.

Enter Filter criteria as follows:

Actions / Field	Description
Save Filters	Click this link to save your filter criteria. A dialog box is displayed so you can name your filter. For easy, repeat access, filters are saved in a "saved filters" box above the link. When you select a saved filter name, the Filter pane fields populate automatically.
Reset	Use this link to clear all entries in the Filter pane.
Quick Search field	Enter any text for your search. For example, enter a host name. You do not need to enter a full name or text string - this field filters on a partial text entry, sometimes 2-3 characters are all you might need.
Status	The status for the agent machine.
Drop-down menu	Default is both enabled and disabled resources. Use the drop-down arrow to choose enabled or disabled only.
Pools field	List one or more pool names, separated by a space. (No spaces are allowed within a pool name). Pool names must be entered exactly - a shorten version of the pool name will not yield a successful result.
Hosts field	List one or more full host names as shown in the Table view "HTTPS & Host" column (each name separated by a space). Entering a partial name will not yield a successful result.
Step Limit drop-down menu	Choose the step limit corresponding to your desired search.
Proxy Agent drop-down menu	Select No if not searching for a proxy agent or Yes if you are trying to locate a proxy agent.

Actions / Field	Description
Filter button	Select this button after you complete the fields/selections to define your filter criteria.

New Resource panel

To define a new standard resource, enter information in the fields as follows:

Field	Description
Name	Enter a unique name for this resource. Do not use "spaces" in a resource name. This is the name used to select the resource for a job step—it need not be the same as the host name for the machine.
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code>
Agent Host Name	Enter the domain name or IP address of the agent machine for this resource. This is the name all other machines in this agent/resource's zone will use to communicate with this agent host.
Agent Port Number	Enter the port number to use when connecting to the agent for the resource, Default port is 7800.
Default Workspace	Enter the workspace name or leave blank to use the local, default workspace. The workspace specification can be overridden at the project, procedure, or step level. If you specify a workspace name here, it will be used as the default for all job steps that run on this resource. See the Workspaces Help topic for more information.
Pool(s)	A space-separated list of arbitrary names indicating the pools associated with this resource. Do not use "spaces" in a pool name.

Field	Description
Host Type	<p>Use the drop-down menu to select the host type:</p> <ul style="list-style-type: none">• Concurrent• Registered <p>Depending on the license installed on the ElectricFlow server, this field may appear.</p> <ul style="list-style-type: none">• If the license on the server is a concurrent resource license, the host type defaults to Concurrent and this field does not appear.• If the license on the server is a registered host license, the host type defaults to Registered and this field does not appear.• If the license on the server is a mixed-mode license (concurrent resources and registered hosts), you must specify the host type when adding a resource.

Field	Description
Default Shell	<p>The name of the shell program used to execute the step's commands on a resource. For example, using <code>sh</code> or <code>cmd /c</code>, the agent saves the command block to a temporary file with a <code>.cmd</code> extension and runs it: <code>sh foo.cmd</code> or <code>cmd /c foo.cmd</code>.</p> <p>If you do not specify a shell on a step, at step run-time the server looks at the resource shell. If a resource shell is not set, the shell line used by the agent is platform dependent:</p> <p>Windows: <code>cmd /q /c "{0}.cmd"</code> UNIX: <code>sh -e "{0}.cmd"</code></p> <p>When you specify a shell (in the step or resource), and omit the <code>cmd-file</code> marker, the agent notices the omission and takes the correct action. For example: a user specifies <code>sh -x</code>. The agent converts this to <code>sh -x "{0}.cmd"</code>.</p> <p>Two alternate forms of shell syntax where ElectricFlow uses a "marker," <code>{0}</code>, as a placeholder for the command file argument:</p> <ul style="list-style-type: none"> • <code><myShell> {0} <potential extra shell args></code> In this example, the command file is not meant to be the last argument in the final command line. For example, <code>mysql -e "source {0}"</code> This shell example runs the <code>mysql</code> command against this step's command containing <code>sql</code>. • <code><myShell> {0} <.file extension> <potential extra shell args></code> In this example, the shell requires the command file to end in an extension other than <code>.cmd</code>. For example, <code>powershell "& '{0}.ps1'"</code> This shell example runs Microsoft PowerShell against this step's command containing PowerShell commands. <p>Notes:</p> <ul style="list-style-type: none"> • When the agent parses the shell, it will parse the extension as everything after <code>{0}.</code> until it sees a space or non-alphanumeric character. • If your script uses International characters (non-ascii), add the following block to the top of your <code>ec-perl</code> command block: <pre>use utf8; ElectricCommander::initEncodings</pre>

Field	Description
Step Limit	<p>The maximum number of steps that can execute simultaneously on this resource—a step limit applies to a particular resource only, not to its underlying host.</p> <p>For example, if you define two resources, resource1 with a step limit of 5 and resource2 with a step limit of 1, both specifying the same host machine, it is possible for a total of 6 steps to execute simultaneously on the underlying host.</p> <p>The Resource Details panel displays this information.</p> <p>Note: Setting the Step Limit to "0" or leaving the field blank defaults to NO step limit.</p>
Artifact Cache Directory	<p>The directory on this resource's agent host from which artifact versions are retrieved and made available to job steps.</p> <p>Enter an absolute path to the resource containing this cache directory.</p>
Zone	<p>The name of the zone where this resource is or will be a member—a zone is a top-level network containing one or more mutually accessible resources.</p>
Repository Names	<p>A "new-line" delimited list of repository names for this resource, if needed. When a step attempts to retrieve artifact versions from an ElectricFlow repository, it queries repositories specified here. If no repositories are specified, it will "fallback" to the default repository.</p>
Connection Type	<p>Use the drop-down menu to select your Connection Type:</p> <ul style="list-style-type: none"> • HTTP - choose this option if you are not using SSL and the agent does not need to be "trusted". • HTTPS - choose this option if you are using SSL, but this agent does not need to be "trusted". • Trusted HTTPS - choose this option if you are using SSL and this agent must be "trusted". A trusted agent is "certificate verified"—the agent verifies the server or "upstream" agent's certificate. <p>Note: For information about the certificate used for trusted agents, see the ElectricFlow-Installed Tools, "eccert" section, Help topic.</p> <p>Agents can be either <i>trusted</i> or <i>untrusted</i>:</p> <ul style="list-style-type: none"> • <i>trusted</i> - the ElectricFlow server verifies the agent's identity using SSL certificate verification. • <i>untrusted</i> - the ElectricFlow server does not verify agent identity. Potentially, an untrusted agent is a security risk.

Field	Description
Enabled	<p>Select this box to enable this resource.</p> <p>When this box is checked, the resource is enabled, which means job steps can be assigned to it. If a job step requests a pool containing this resource, the step executes on a resource in that pool.</p> <p>If disabled, and this is the only resource that satisfies a particular job step, the step's execution is delayed until the resource is re-enabled. If a resource is disabled while job steps are running on it, the running steps continue to completion but no new steps are assigned to that resource.</p>

Click **OK** to save your settings and see your new resource listed in the table.

New Proxy Resource panel

Reminder - proxy agent vs. proxy target:

- Proxy *agent*—This is an agent on a supported Linux or Windows platform, used to proxy commands to an otherwise unsupported platform.
- Proxy *target*—This is a machine on an unsupported platform that can run commands via an SSH server.

To define a new proxy resource, enter information in the fields as follows:

Field	Description
General	
Name	Enter a unique name for this resource. Do not use spaces in a resource name. This is the name used to select the resource for a job step; it does not need to be the same as the host name for the machine.
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code>
Proxy Agent Host Name	Enter the domain name or IP address of the proxy agent machine corresponding to this resource.
Proxy Agent Port Number	Enter the port number to use when connecting to the agent for the resource.

Field	Description
Default Workspace	<p>Enter the workspace name for leave blank to use the local, default workspace.</p> <p>If you specify a workspace here, it will be used as the default for all job steps that run on this resource. See the Workspaces Help topic for more information.</p>
Pool(s)	<p>Enter a space-separated list of arbitrary names, indicating the pools associated with this resource. Do not use "spaces" in a pool name.</p>
Host Type	<p>Use the drop-down menu to select the host type:</p> <ul style="list-style-type: none"> • Concurrent • Registered <p>Depending on the license installed on the ElectricFlow server, this field may appear.</p> <ul style="list-style-type: none"> • If the license on the server is a concurrent resource license, the host type defaults to Concurrent and this field does not appear. • If the license on the server is a registered host license, the host type defaults to Registered and this field does not appear. • If the license on the server is a mixed-mode license (concurrent resources and registered hosts), you must specify the host type when adding a resource.
Step Limit	<p>Specify the maximum number of steps that can execute simultaneously on this resource—a step limit applies to a particular resource only, not to its underlying host.</p> <p>For example, if you define two resources, <code>resource1</code> with a step limit of 5 and <code>resource2</code> with a step limit of 1, both specifying the same host machine, it is possible for a total of 6 steps to execute simultaneously on the underlying host.</p>
Zone	<p>The name of the zone where this resource will be a member.</p>

Field	Description
Connection Type	<p>Use the drop-down menu to select your Connection Type:</p> <ul style="list-style-type: none"> • HTTP - choose this option if you are not using SSL and the agent does not need to be "trusted". • HTTPS - choose this option if you are using SSL, but this agent does not need to be "trusted". • Trusted HTTPS - choose this option if you are using SSL and this agent must be "trusted". A trusted agent is "certificate verified"—the agent verifies the server or "upstream" agent's certificate. <p>Note: For information about the certificate used for trusted agents, see the ElectricFlow-Installed Tools, "eccert" section, Help topic.</p> <p>Agents can be either <i>trusted</i> or <i>untrusted</i>:</p> <ul style="list-style-type: none"> • <i>trusted</i> –The ElectricFlow server verifies the agent's identity using SSL certificate verification. • <i>untrusted</i> –The ElectricFlow server does not verify agent identity. Potentially, an untrusted agent is a security risk.
Enabled	<p>Select this box to enable this resource.</p> <p>When this box is checked, the resource is enabled, which means job steps will be assigned to it. If a job step requests a pool containing this resource, the step executes on a resource in that pool.</p> <p>If disabled, and this is the only resource that satisfies a particular job step, the step's execution is delayed until the resource is re-enabled. If a resource is disabled while job steps are running on it, the running steps continue to completion but no new steps are assigned to that resource.</p>
Proxy Target	
Proxy Target Host Name	Enter the domain name or IP address of the proxy target machine corresponding to this resource.
Proxy Target Port Number	Enter the port number to use when connecting to the proxy target host or leave blank to use the default. The default port is 22, the SSH server port.

Field	Description
Proxy Target Default Shell	<p>The name of the shell program used to execute commands on this [the proxy target] resource—may be overridden at the step level.</p> <p>This shell will be used to execute the step's commands on a resource. For example, using <code>sh</code> or <code>cmd /c</code>, the agent saves the command block to a temporary file with a <code>.cmd</code> extension and runs it: <code>sh foo.cmd</code> or <code>cmd /c foo.cmd</code>.</p> <p>If you do not specify a shell on a step, at step run-time the server looks at the resource shell. If a resource shell is not set, the shell line used by the agent is platform dependent:</p> <p>Windows: <code>cmd /q /c "{0}.cmd"</code> UNIX: <code>sh -e "{0}.cmd"</code></p> <p>When you specify a shell (in the step or resource), and omit the <code>cmd-file</code> marker, the agent notices the omission and takes the correct action. For example: a user specifies <code>sh -x</code>. The agent converts this to <code>sh -x "{0}.cmd"</code>.</p> <p>Two alternate forms of shell syntax where ElectricFlow uses a "marker," <code>{0}</code>, as a placeholder for the command file argument:</p> <ul style="list-style-type: none"> • <code><myShell> {0} <potential extra shell args></code> In this example, the command file is not meant to be the last argument in the final command line. For example, <code>mysql -e "source {0}"</code> This shell example runs the <code>mysql</code> command against this step's command containing <code>sql</code>. • <code><myShell> {0} <.file extension> <potential extra shell args></code> In this example, the shell requires the command file to end in an extension other than <code>.cmd</code>. For example, <code>powershell "& '{0}.ps1'"</code> This shell example runs Microsoft PowerShell against this step's command containing PowerShell commands. <p>Notes:</p> <ul style="list-style-type: none"> • When the agent parses the shell, it will parse the extension as everything after <code>{0} .</code> until it sees a space or non-alphanumeric character. • If your script uses International characters (non-ascii), add the following block to the top of your <code>ec-perl</code> command block: <pre>use utf8; ElectricCommander::initEncodings</pre>

Field	Description
Proxy Customizations	This is proxy-specific customization data. Default is "none". See the "ElectricFlow-Installed Tools" Help topic (ecproxy section) for more information.

Click **OK** to save your settings.

Edit Resource panel

All of the fields on this panel are the same as those on the New Resource panel:

- All information you previously provided to create this resource is filled-in for you.
- You can change any information already supplied.
- You can add information in any blank fields.

Click **OK** to save your modifications.

IMPORTANT: If you want to change the Connection Type, using the UI by itself is insufficient. You must also use the `ecconfigure` utility to change the protocol. Run: `ecconfigure --agentProtocol [http|https]`

Edit Proxy Resource panel

All of the fields on this panel are the same as those on the New Proxy Resource panel:

- All information you previously provided to create this resource is filled-in for you.
- You can change any information already supplied.
- You can add information in any blank fields.

Click **OK** to save your modifications.

Resource Details panel

This panel displays how this resource is configured with the ElectricFlow server. Some of the information supplied is linked to its source. For example, the zone name for this resource is a link to its Zone Details panel.

- The name of the resource is shown below the panel title.
- The resource description is shown under the resource name.
- Links at the top of the panel:
 - Use the **View Usage** link to go to the Job Step Search Results page.
 - Use the **Edit** link to modify the resource specifications.
 - Use the **Access Control** link [at the top of the panel] to specify or view access privileges for this resource.

- The tabs:
 - General - by default, the panel opens to display the "general" resource configuration.
 - Properties - select this tab to add properties to this resource or see existing resource properties if already configured.

Custom Resource Properties

You can define custom properties for any ElectricFlow object. For example, if configuration information varies from resource to resource, use custom properties to store this information and then reference it from procedure steps where it is needed.

For example, if you have a pool of test machines with test hardware in a different location on each machine, you could define a property named "testLocation" on each resource, then pass this property to procedure steps using a reference such as `$/myResource/testLocation`.

Switching a non-trusted agent to trusted

If you have upgraded to ElectricCommander 4.2 or later, none of your existing agents were converted to "trusted" during the upgrade process. By default, the *local* agent is not trusted during a new or upgrade installation.

After an upgrade, if you want to use a trusted configuration, any **pre-existing** agent/host machines must be manually switched to "trusted".

Manually switching a non-trusted agent to trusted status

Perform the following steps on the agent machine to change the agent status to trusted in your environment and in the customer's site.

1. Enter `ectool --server <COMMANDER_SERVER_IP> login admin` to log into the server and save the session ID.
2. Enter `eccert --server <COMMANDER_SERVER_IP> initAgent --remote --force` to install the agent with a self-signed certificate that needs to be overwritten.
3. Restart the agent.


If you do not restart the agent, it tries to use a previously cached certificate if one exists. The old certificate is invalid because a new one was just created.

4. In the ElectricFlow UI, ping the agent.

For information about the ElectricFlow Certificate Authority (CA) and the certificates configured in ElectricFlow Server and ElectricFlow Agent installations, see [ElectricFlow Installed Tools](#) on page 602.

Install or Upgrade Remote Agents



Select this icon, , to install or upgrade host machines you want to use as agent machines for your ElectricFlow resources. A dialog box opens where you enter information about the hosts to install or upgrade.

- If no hosts are selected in the Resources Table view, you will be installing host names you enter.
- If host names are listed already in the Resources Table view, selected one or more to perform an upgrade operation.

Notes:

- The dialog screens you see for an Install operation will be somewhat different from those for an Upgrade operation, depending on the options you choose.
- You can install or upgrade Linux, Solaris, Solaris x86, HP-UX, or MacOS hosts.
- When you install the agent software on a Solaris host, install it in a directory path that has less than 70 characters. .

Prerequisites:

- You must have an artifact repository server installed.
- The target machines must be running the SSH daemon.
- The user you select must be set up for passwordless *sudo* privileges.
For example, in the `/etc/sudoers` file, you must add "`<username> ALL=(ALL) NOPASSWD:ALL`"

IMPORTANT: If the EC-AgentManagement plugin is *not* installed, the remote install/upgrade option will not be available and you cannot install/upgrade agents remotely from the automation platform user interface. During a new ElectricFlow installation, this plugin is installed automatically.

- For each remote agent machine you want to upgrade, make sure PasswordAuthentication=yes in `/etc/ssh/sshd_config`.
Example:

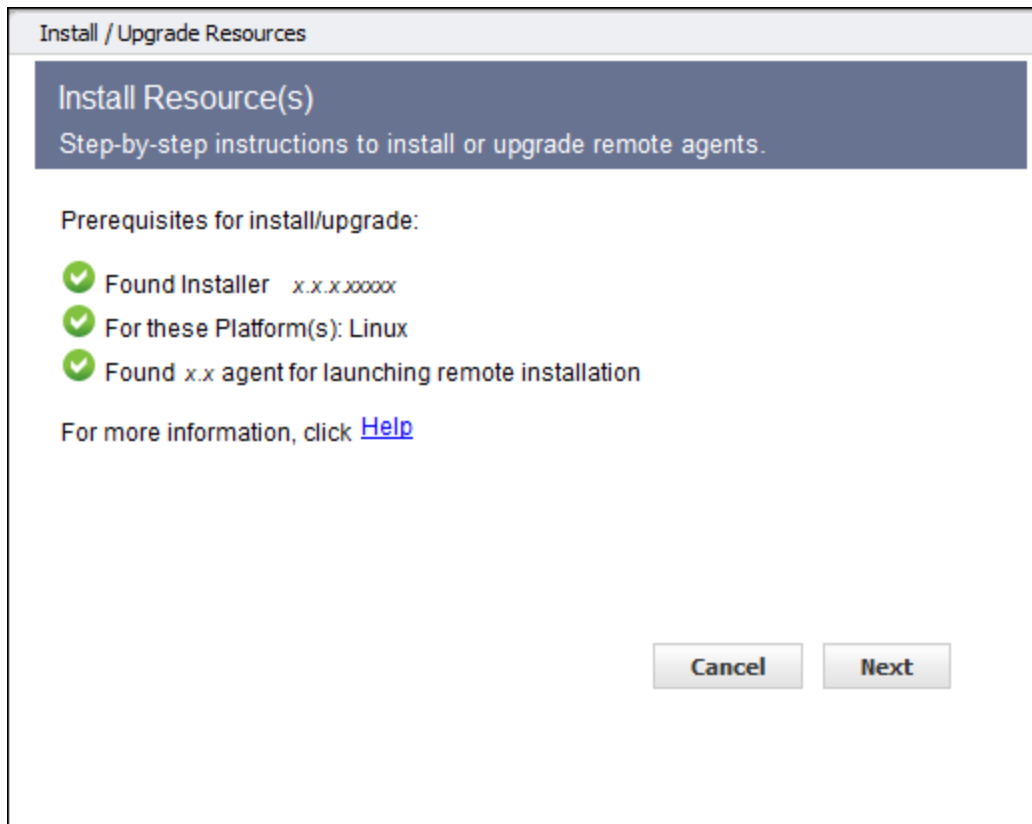
```
# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes
```


PasswordAuthentication=no is the default on SUSE platforms, so you must edit `/etc/ssh/sshd_config`. Restart the sshd service after changing the value.
- On Unix, SSH has the following additional prerequisite:
 - The user account on the target machine must have password-less sudo configured for running the installer with root privileges.

Using the Install or Upgrade Remote Agents dialog

The ElectricFlow server runs several verification checks before you begin an Install/Upgrade operation, and when you see the following screen with all green "check mark" icons, you can proceed.

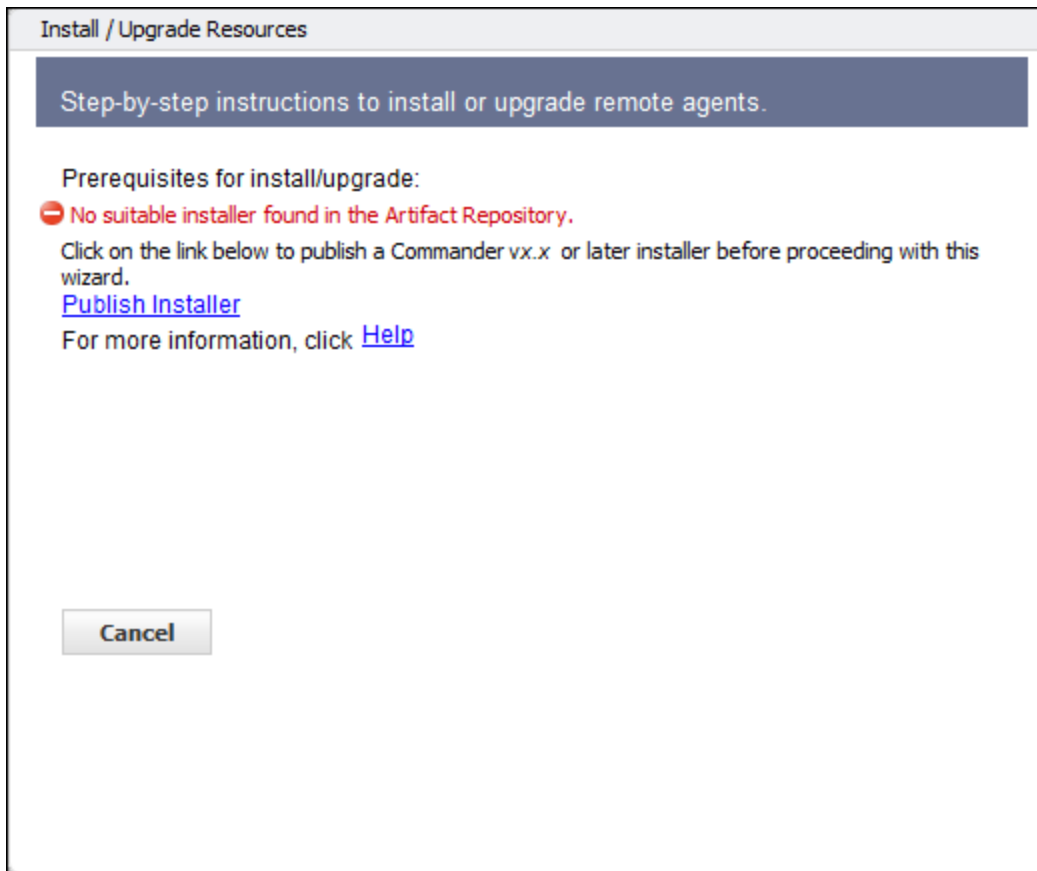
Note: Depending on your selections, you will see variations in some to the following screen examples.



Click **Next** to continue.

However, if the plugin is installed but an ElectricFlow 4.2 or later installer is not found in the artifact repository, you will see an error screen similar to the example below.

Click the **Publish Installer** link.



After clicking the **Publish Installer** link, the Run Procedure – Publish Installer page opens.

Enter information in the fields as follows:

Parameters

fromDir—Enter the path to the directory containing the installer on the ElectricFlow server machine. For example, `/var/tmp` for Linux.

platform—Use the drop-down menu to make a selection.

repository—Use `default` to use the repository installed during the ElectricFlow installation, or type-over `default` and enter a different repository name.

resource—Use the name of the resource to be used by this procedure.

version—Enter the build version for your ElectricFlow install, for example, "5.0.0.56390". Your build version can be obtained from the name of the ElectricFlow installer you downloaded from the Electric Cloud FTP site.

Advanced

(Not required to complete the Publish Installer run procedure)

When the Publish Installer procedure runs, you can see the job status on the Job Details page.

For verification:

Go to the Artifacts page > select `com.electriccloud:installer`, which displays the Artifact Details page. This page displays all of your published installers.

Return to the Resources page and begin again...

Assuming you now have all green "check mark" icons on the first dialog screen, and you clicked **Next**, the following dialog screen is displayed.

Install / Upgrade Resources

Target Options
Supply list of hostnames to install

Zone:

Install to Host:

Name Templates:

Template:

Enter information in the fields as follows:

Zone—If you have not defined additional zones, the `default` zone is used and you will not see this field. If you have multiple zones defined already, use the drop-down menu to choose the zone where you are installing or upgrading agents. *You can install or upgrade agents to only one zone at a time.* A functioning

gateway must exist before an agent can be installed into the non-default zone. So installing the first agent into a zone must be done manually.

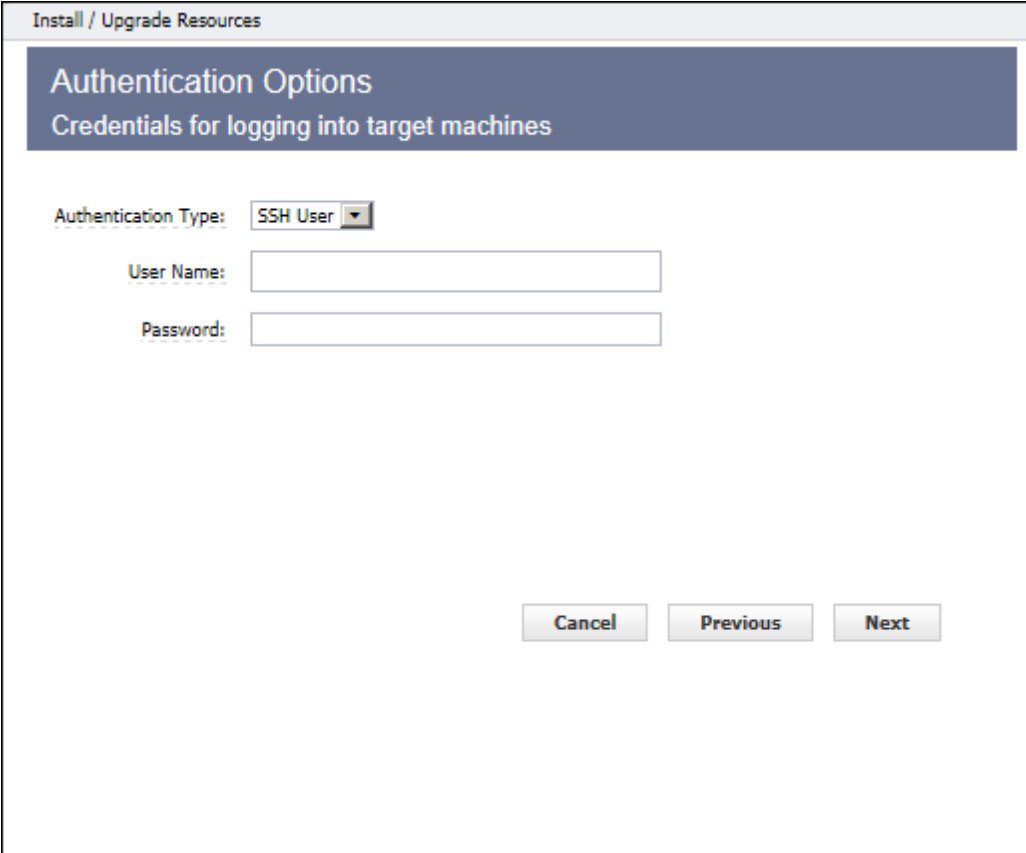
Install to Host—(required) Enter one or more host names or IP addresses in a space-separated list. Individual host names *cannot* contain spaces. If this is an upgrade operation, host names or IP addresses are pre-populated in this field from the resources you selected for upgrade on the Resources page - Table view, first column.

Name Templates—Use the drop-down menu to choose a simple template for setting the names of newly created resources.

Template—You can edit this field only if you selected Custom in the Name Template drop-down menu. This field undergoes property expansion in a global context and is scanned for special tokens {name} and {counter}.

Driving Resource—This resource performs all actions for installing or upgrading a target host on behalf of the server.

Click **Next** to continue to the next dialog screen.



Install / Upgrade Resources

Authentication Options
Credentials for logging into target machines

Authentication Type: SSH User

User Name:

Password:

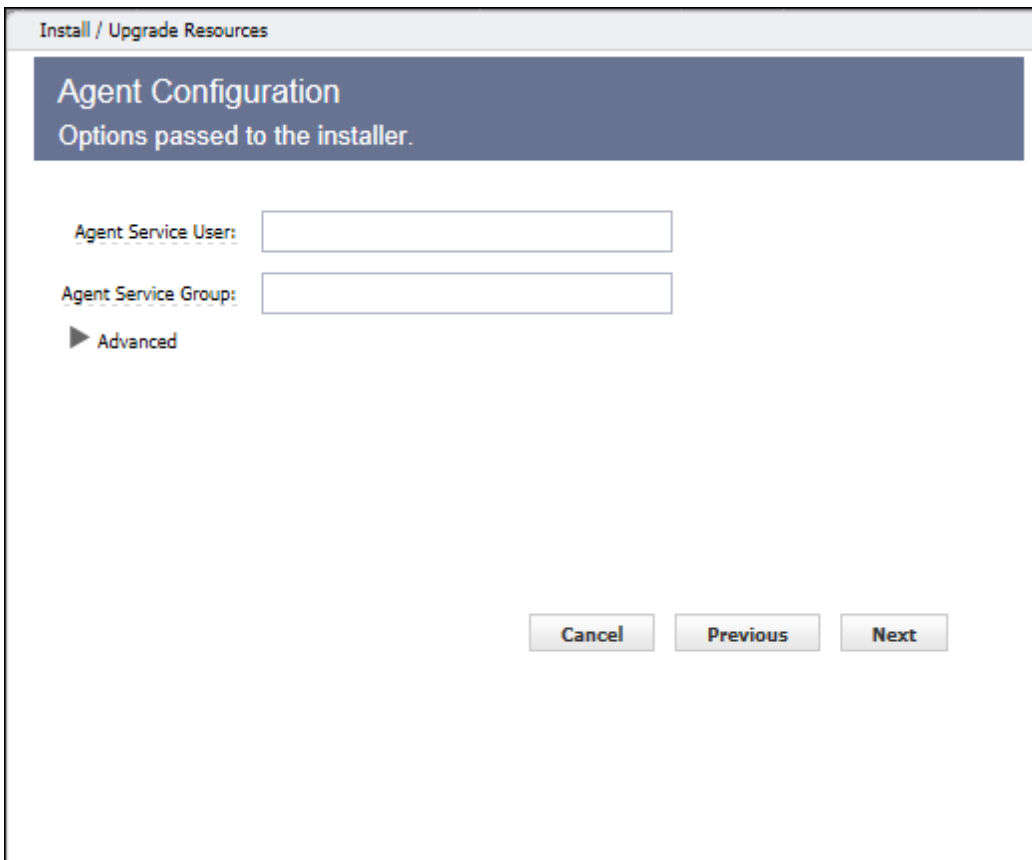
Cancel Previous Next

Enter information in the fields as follows:

Authentication Type - use the drop-down menu to choose SSH User, SSH Key, or WMI:

- If SSH User -
 - User Name - (required) Enter a user name.
 - Password - (required) Enter a password for the user name.
- If SSH Key -
 - User Name - (required) Enter a user name.
 - Public Key Path - (required) The path to the SSH public key file.
 - Private Key Path - (required) The path to the SSH private key file.
 - Passphrase - (optional) The passphrase for unlocking the private key file.
- If WMI -
 - No additional information is needed

Click **Next** to continue to the next dialog screen.



The image shows a screenshot of a software installation window titled "Install / Upgrade Resources". Inside the window, there is a section titled "Agent Configuration" with the subtitle "Options passed to the installer." Below this, there are two input fields: "Agent Service User:" and "Agent Service Group:". Below these fields is a link labeled "Advanced" with a right-pointing triangle icon. At the bottom right of the window, there are three buttons: "Cancel", "Previous", and "Next".

Note: You will not see this screen if you are *upgrading* hosts.

For installing hosts, enter information in the fields as follows:

Agent Service User - (required) the user the ElectricFlow agent "runs as".

Agent Service Group - (required) the group the ElectricFlow agent "runs as".

Selecting "Advanced" is optional - Select this option to define non-default values.

- If you did not select Advanced, click **Next** to continue (and skip the next dialog screen example).
- If you selected Advanced, see the next dialog screen example.

The screenshot shows a dialog box titled "Install / Upgrade Resources" with a sub-header "Agent Configuration" and the text "Options passed to the installer." Below this, there are input fields for "Agent Service User:" and "Agent Service Group:". A section labeled "Advanced" is expanded, showing input fields for "Agent Port:", "Install Directory:", and "Data Directory:", along with a "Trusted Agent:" checkbox. At the bottom right, there are three buttons: "Cancel", "Previous", and "Next".

Advanced options: (optional)

Agent Port - Specify the listener port for the agent.

Install Directory - Specify the path to the install directory.

Data Directory - Specify the path to the directory where your modified ElectricFlow files are stored (configuration and log files).

Trusted Agent - Select this check box and all agent machines being installed or upgraded will be "trusted".

For more information, see [Switching a non-trusted agent to trusted](#) on page 832.

Click **Next** to continue to the summary dialog.

Your summary dialog will be similar to the following example, but it will contain the values you specified.

Note: If you are installing or upgrading more than 5 hosts, you will see a quantity number and NOT a space-separated list of host names

Install / Upgrade Resources

Ready to Install

Review the settings before starting the install.

Number of Host(s):

gerrit jira-test linplugin1

Zone:

alpha

Template:

Hostname/IP

UserName:

build

Password:

[PROTECTED]

Agent Service User:

qe

Agent Service Group:

build

Trusted Agent:

Untrusted

Select Finish to run the install.

Cancel

Previous

Finish

Click **Finish** to start the install or upgrade and the Job Details page is displayed.

When install/upgrade is complete, you can return to the Resources page to see your newly installed or upgraded hosts listed in the Table or Grid view.

To verify a resource version, click a resource name (in the Resources page Table view) to open the Resource Details panel for that resource.

Resource Pools

This page displays all resource pools currently available to ElectricFlow.

- To create a new resource pool, click the **Create Resource Pool** link.
- To find an existing resource pool, use the **New Search** link.

Column descriptions

Column Name	Description
Pool Name	This is the name you supplied for this resource pool. Click on a Pool Name to go to the Resource Pool Details page.
Enabled	If this box is "checked", this resource pool is enabled.
Auto Delete	If this column is selected with a check mark, the pool will be deleted when the last resource is deleted or removed.
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code>
Actions	<p>Use the Copy link to make an exact duplicate of an existing resource pool, then select the copy to go to the Resource Pool Details page to change the resource pool name or add or remove resources from the pool.</p> <p>Use the Delete link to delete the resource pool on the same row.</p> <p>Note: Deleting a resource pool does not delete the resources referenced by the pool.</p>

Resource Pool - create new or edit existing pool

To create a new resource pool

Enter information in the fields as follows:

Field Name	Description
Pool Name	Enter a name for the resource pool. This name must be unique within a list of multiple resource pool names and cannot be the same as any resource name.
Enabled	Select this check box to enable the resource pool.

Field Name	Description
Type	<p>A resource can be <i>static</i> or <i>dynamic</i>.</p> <ul style="list-style-type: none"> A static resource is part of your system or network (not in the cloud), such as a server, database, or agent machine. A dynamic resource is a cloud resource that can be provisioned and later spun up on-demand when an application is deployed.
# Resources	Number of resources in the resource pool.
Resource Template Name	<p>The name of the resource template to which the resource is applied.</p> <p>Note: Only dynamic resources can be applied to a resource template.</p>
Auto Delete	<p>If you select this check box, the pool is deleted when the last resource is deleted or removed.</p> <ul style="list-style-type: none"> The <code>autoDelete</code> property flag is automatically set to "true" if the pool was created on the Create Resource page, as a side-effect of the resource API calls. By default, resource pools created explicitly on the Create Resource Pool page, have the <code>autoDelete</code> flag set to false—the checkbox is "unchecked".
Description	<p>A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code></p>
Actions	<p>Actions you can perform on the resource pool, including:</p> <p>Copy—Creates a new resource by copying the this resource pool.</p> <p>Delete—Deletes this resource pool.</p> <p>Tear Down—Removes the dynamic resource pool from a dynamic environment.</p> <p>Track Changes—Opens the Change History for the resource pool.</p>

Click **OK** to save the information to create a new resource pool. You will see your new resource pool listed on the Resource Pools page.

Note: Resource pools cannot be nested, which means if the name of a resource pool is added to a resource pool, it will be ignored.

A resource pool can override the default algorithm by providing a JavaScript fragment in the Ordering Filter (`orderingFilter`) intrinsic property. The JavaScript environment for this fragment is:

- A variable named `resourceList` – an array of the resources in the pool, sorted as described above
- A variable named `resourcePool` – the pool object
- The job step being scheduled is configured as the context object for property resolution

The evaluation result is expected to be an array of resources, in the order they should be considered by the scheduler.

Before evaluation, the JavaScript fragment is wrapped as a function, to make it possible to use the `return` statement in the script.

It is possible to return any resources in the result, not just those grouped in a pool, so this feature enables true dynamic resource selection. Resources are still subject to the "usable" criteria. See the "Unusable resources" section at the end of this Help topic.

Examples for the Ordering Filter field:

- Sort by resource name in descending order:

```
return resourceList.sort(function(a,b) {
return b.resourceName.localeCompare(a.resourceName);
});
```

- Ignore the default resources and return different resources:

```
var result = new Array();
result[0] = resources["resource1"];
result[1] = resources["resource2"];
return result;
```

To edit an existing resource pool

Use this page to modify any specifications for the resource pool.

Click **OK** to save the information and update the resource pool.

Unusable resources

Resources, whether in a pool or not, are not considered usable if any of the following are true:

- The resource does not exist.
- The resource is disabled.
- The resource does not have a host name.
- The agent where the resource is associated is not reachable.
- The number of steps running on the resource is equal to the resource step limit.
- The resource is exclusive to a different job using the job `exclusiveMode`.
- The resource is exclusive to a step in a different job using the step or call `exclusiveMode`.

- The resource is exclusive to a different step in the same job using the call `exclusiveMode` and the other step is not an ancestor of the step.
- The resource is exclusive to a different step in the same job using the step `exclusiveMode`.
- The job step does not have execute privileges on the resource.
- The resource assignment exceeds the license limit (if any).

Resource Pool Details

This page displays information for the resource pool you selected, including a summary of the specifications supplied to create the resource pool, current resources assigned to this pool, and the ability to add or remove resources.

Links and actions at the top of the page

- **Edit** - use this link to go to the Edit Resource Pool page.
- **Access Control** - use this link to go the Access Control page for this resource pool.
- The "star" icon allows you to save this information to your Home page.
- The "curved arrow" icon returns you to the Resource Pools page.
- **Pagination** - Use the "previous" and "next" arrow icons to view the previous or next project. The numbers between the arrow icons display the number of projects you can view and the first number indicates which project [in the list] you are viewing.

The "tabbed" sections

The tabs at the top of the table allow you to select the type of information you want to see. The Resource Pool Details page opens with the Resources tab highlighted, so the first table you see is the Resources table. See the following screen example.

This table displays the resources currently included in this pool. The name of the pool ("custom" in our example) follows the Resource Pool Details page title.

Resource Pool Details / default									
Description: Default resource pool containing local agent created during installation. Ordering Filter: Auto Delete: no Enabled: yes									
Resources Properties									
Resource Name	Status	Enabled	Description	Host	Proxy Agent	Version	HTTPS	Add Resource to Pool	Step Limit
local			Local resource created during installation.	flow-demo-uat.electric-cloud.com		6.0.0.94411			1/1

The summary section at the top of the table contains information previously defined when the pool was created:

Description - The text previously supplied for this object when it was created.

Ordering Filter - The Javascript ordering filter (see the [New Resource Pool](#) Help topic for more information) or "empty" if no filter was applied.



Auto Delete - When "yes" is specified, the pool will be deleted when the last remaining resource is removed or deleted from the pool.

Enabled - "yes" specifies this pool is enabled for use.

Links at the top of the table

Add Resources to Pool - Click this link to see a pop-up dialog to enter a resource name to add to this pool. You can type a name or select from a list of existing resource names. If other resources are already configured, you will see a list from which to select a resource. Click **OK** to add new resources to the pool.

Column descriptions

Column Name	Description / Actions
Resource Name	The name of the resource. Click on a resource name to go to the Edit Resource page.
Status	Indicates the current status of the resource.
Enabled	Indicates whether the resource is enabled. If a resource is disabled, no job steps will be assigned to it. ElectricFlow will use other resources within the pool to satisfy requests for the pool.
Description	The text previously supplied for this object when it was created.
Host	Steps assigned to run on this resource will use this host.
Proxy Agent	The name or IP address of the proxy agent machine.
Version	The version of the ElectricFlow agent installed on this resource.
HTTPS	A "checkmark" indicates HTTPS was selected when this resource was created.
Step Limit	The maximum number of steps that can execute simultaneously on this resource.
Actions	<div style="text-align: center;">  </div> <p>Ping - Use this link to check the status of the resource.</p> <div style="text-align: center;">  </div> <p>Remove - Use this link to remove a resource from this pool.</p>


Properties tab

This tab provides a table listing all properties for the resource pool and includes the following functionality:

Links at the top of the table

- To create a new property for this resource pool, click the **Create Property** or **Create Nested Sheet** link.
- To view or change privileges on the property sheet, click the **Access Control** link.

Column descriptions

Column Name	Description / Action
Property Name	The name of the property. Click on a property name to edit that property.
Value	Indicates the value assigned to this property
Description	A text description previously supplied for your reference only.
Actions	 Delete - Use this link to delete this property.

Zones

This page displays zones currently available to ElectricFlow and provides other zone operations—create, edit, delete, and so on.

- A zone is a way to partition a collection of agents to secure them from use by other groups. For example, you might choose to create a developers zone, a production zone, and a test zone—agents in one zone cannot directly communicate with agents in another zone.
- A *default* zone is created during the ElectricFlow installation. The server implicitly belongs to the default zone, which means all agents in this zone can communicate with the server directly (without the use of a gateway).
- Each zone can have one or more "gateway agents", which you define. Gateway agents are used for communication from one zone to another zone. For more information, see the [Gateways](#) Help topic.
- Every agent, and all resources defined on that agent, can belong to one zone only.
- Within a zone, agents can be either *trusted* or *untrusted*.
 - *trusted* - the ElectricFlow server verifies the agent's identity using SSL certificate verification.
 - *untrusted* - the ElectricFlow server does not verify agent identity. Potentially, an untrusted agent is a security risk.

Links and actions at the top of the table

- To create a new zone, click the **Create Zone** link.
- The "star" icon allows you to save this zone information to your Home page.

Column descriptions

Column Name	Description / Actions
Name	The name supplied when this zone was created. Click on a Zone Name to see the Zone Details panel for that zone.
Descriptions	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code>
Actions	<p>Delete - Use this link to delete the zone on the same row.</p> <p>Note: Deleting a zone does not delete resources in the zone. All agents (and their respective resources) move to the default zone if their current zone is deleted. The <i>default</i> zone cannot be deleted.</p>

Zone Details panel

This panel displays the zone name and description for the zone you selected.

Links and actions at the top of the table

- **Resources**—Use this link to go to the Resources page.
- **Edit**—Click this link to display the Edit Zone panel.
On the Edit Zone panel, you can change the zone name ,or you can add or change the zone Description.
Click **OK**after making any changes.
- **Access Control** - Use this link to go to the Access Control page for this zone.
 - Current access privileges are displayed, if any.
 - Add the access control privileges you need for this zone.

Select the Properties tab to view any properties created for this zone, or use the "create" property links to create properties for this zone.

Creating a new zone

Enter information in the fields as follows:

Field Name	Description
Name	Enter a name of your choice for this zone. The name must be unique among other zone names.
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code> .

Click **OK** to see your new zone displayed in the table.

To add resources to this zone, go to the Resources page.

Access Control notes

A zone inherits privileges from the ZonesAndGateways ACL. See the [Access Control](#) ("Server Objects" section) Help topic for more information.

Resource and Resource Pool continue to inherit from Resources privileges. To create a resource, you must have modify privileges on Resources (this has not changed) and you must have modify privileges on the zone. In addition, to move a resource from one zone to another, you must have modify privileges on both zones and the resource you want to move.

Gateways

To communicate with a resource, workspace, or artifact repository server in another zone, a "gateway" must be created. A gateway object contains two resource (agent) machines, for example, GatewayResource1 and GatewayResource2—each configured to communicate with the other. One gateway resource resides in the *source* zone and the other in the *target* zone. A gateway is bidirectional and informs the ElectricFlow server that each gateway machine is configured to communicate with its other gateway machine (in another zone).

If your company requires the added security of a firewall between zones, gateway agents can be configured to communicate with/through the firewall.

This page displays all gateways currently defined in ElectricFlow and provides other gateway operations—create, edit, delete, and so on.

- A firewall between zones - A gateway resource can be configured to communicate with an intermediary firewall in its path as a proxy to communicate with its peer on the other side of the gateway.
- Each gateway records the host/port combination each gateway agent/resource must use to communication with its peer on the other side of the gateway.

- Multiple gateways can be defined for a zone if required.
For example, you may have multiple resources in zoneA that need to communicate with each other, but some of those resources also need to communicate with zoneB, while others need to communicate with zoneC only. In this scenario, zoneA would require two gateways—one to zoneB and one to zoneC.
- One resource can participate in multiple gateways.
For example, assume we have 3 zones, zone1, zone2, and zone3, each created to contain agent/resource machines for a different, specific purpose (production, testing), but we want to share or pass data from a resource in zone1 to another resource in zone2 or zone3:
 - We need two gateways:
 - Gateway1 connects ResourceA in zone1 to ResourceC in zone3
 - Gateway2 connects ResourceA in zone1 to ResourceB in zone2
 - With this gateway-resource configuration, ResourceA can communicate directly with zone2 or zone3.

Links and actions at the top of the table

- To create a new gateway, click the **Create Gateway** link.
- The "star" icon allows you to save this gateway information to your Home page.

Column descriptions

Column Name	Description / Action
Name	The gateway name specified when this gateway was created. Click on a gateway name to see the Gateway Details panel for that gateway.
Enabled	A "checkmark" in the box indicates this gateway is enabled.
Resource 1	The first of two resources required to create a gateway.
Host 1	The name Resource 2 uses to communicate with Resource 1. If "blank", the <i>Agent Host Name</i> attribute in Resource 1's definition is used at runtime.
Resource 2	The second of two resources required to create a gateway.
Host 2	The name Resource 1 uses to communicate with Resource 2. If "blank", the <i>Agent Host Name</i> attribute in Resource 2's definition is used at runtime.
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code>
Actions	Delete - Use this link to delete the gateway on the same row.

Gateway Details panel

This panel displays properties and access control privileges assigned to this gateway.

Select the Properties tab to see any existing properties or to create properties for this gateway.

Links and actions at the top of the panel

- **Edit** - Click this link to display the Edit Gateway panel.
On the Edit Gateway panel, you can change the gateway name or add/change the gateway Description.
- **Access Control** - Click this link to go to the Access Control page to set access privileges for this gateway.

Create Gateway panel

Enter the following information:

Field Name	Description / Action
Name	Enter a name of your choice for this gateway. The name must be unique among other gateway names.
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code> .
Enabled	Select this box to enable the gateway.
Resource 1	The name of your choice for the first of two required gateway resources. Do not include "spaces" in a resource name.
Host 1	The agent host name where <i>Resource1</i> resides. This host name is used by <i>Resource2</i> to communicate with <i>Resource1</i> . Leave this field blank to use the host name from <i>Resource1</i> 's definition.
Port 1	The port number used by <i>Resource1</i> - default is to the port number used by the resource.
Resource 2	The name of your choice for the second of two required gateway resources. Do not include "spaces" in a resource name.

Field Name	Description / Action
Host 2	The agent host name where <i>Resource2</i> resides. This host name is used by <i>Resource1</i> to communicate with <i>Resource2</i> . Leave this field blank to use the host name from <i>Resource2</i> 's definition.
Port 2	The port number used by <i>Resource2</i> - default is to the port number used by the resource.

Click **OK** to see your new gateway displayed in the table.

Edit Gateway panel

This panel is populated with previously supplied information to define the gateway.

You can change any existing specifications or add new information.

Click **OK** to save your changes.

Access Control note

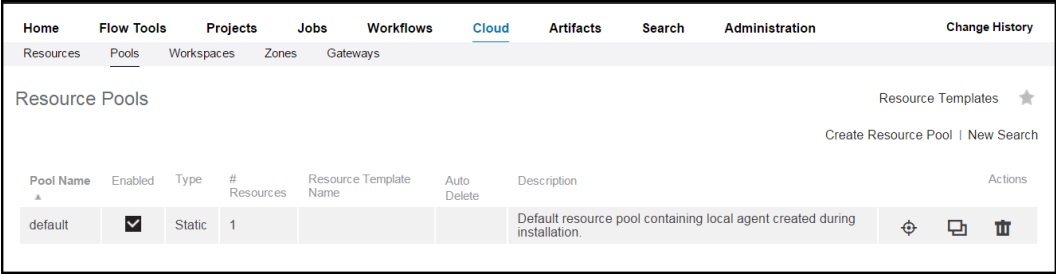
A gateway inherits privileges from the ZonesAndGateways ACL. See the [Access Control](#) ("Server Objects" section) Help topic for more information.

Accessing the Resource Templates in the Automation Platform

In automation platform UI:

1. Go to **Cloud > Pools**.

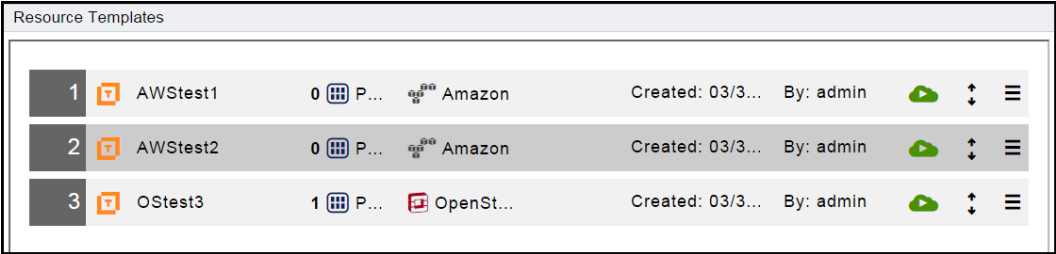
Example:




2. Click **Resource Templates**.

The **Resource Templates** list opens.

Example:



3. (Optional) To view details about a resource template, choose a template and click the **Menu** button () for it.

The Context menu opens.

4. Click one of these options:

- **Details**–The **Edit Resource Template Details** dialog box opens.
The tabs that appear in the dialog box depend on the cloud provider specified in the resource template.
- **Properties**–The **Properties** dialog box opens.
You can view and edit the properties that apply to the resource template.
- **Access Control**–The Access Control page opens.
- **Track Changes**–The Change History for the resource template opens.

- To provision the resource template, choose a resource template and click the **Provision** button.

The **Provision Cloud Resources** dialog box opens.

Example:

- Enter the resource pool name, an optional description, and the number of resources in the pool, and then click **OK**.

Example:

You go to the Job Details page in the automation platform.

Define Search

The page allows you to search for any supported object type.

Enter information in the fields as follows:

Field Name	Description
Number of Results	Default is 200. Change this number to see a greater or lesser search result set.
Results Per Page	Choose the number of search results you would like to see. Default is 20.

Field Name	Description
Object Type	Use the drop-down menu to select the object you wish to search.
Intrinsic Filters	<p>Click the Add Intrinsic Filter link one or more times as needed to further define the sort criteria for your search.</p> <ul style="list-style-type: none"> • Use the drop-down arrow to select an intrinsic filter argument. These arguments are properties. For a description of any of these properties, see the Properties Help topic. • Use the drop-down arrow to select an operator for the search. • To narrow your search, enter an actual object name. For example, if you selected Project Name to begin your search, you might want to enter the actual project name in this field or the first few letters if you have multiple projects named in a similar pattern.
Custom Filters	Click the Add Custom Filter link one or more times as needed to enter custom sort criteria to further define your object search. Type-in the object name to define your search criteria, then select your search operator.

Click **OK** after filling-in the fields and to go to the Search Results page.

Notes:

1. Only the first 450 characters of a property string are searchable. For example, if your property value happens to contain 1500 characters and the information you are trying to find is 800 characters into the property string, the search mechanism will not find it.
2. When defining a search on this page, property path references are **not** allowed in search parameters. However, property path references are allowed in the Home page Quick View filters.

Search Results

This page displays the results of the search you defined.

- If the results were not as expected, click the **Edit Search** link to modify your search criteria.
- Click the **New Search** link to define a new object you need to find.
- Click the **Save Filter** link if you want to save this filter for future use in a report. In the pop-up screen, enter a name for the filter and click **OK**. The Saved Filter will be stored as a property in your chosen project.
- To continually monitor the status of your found object, click the **Refresh Search** link.

Notes:

If you need to refer to this page on a regular basis, mouse-over the "star" icon to add this page to the Shortcut section on your Home page.

If you use RSS, an active RSS icon is provided in Windows Explorer on this page for your convenience. If you use Firefox, click **Bookmarks** > Subscribe to this page to display the feed. You can then add the feed URL to a viewer of your choice.

Server

This page displays overall information about the ElectricFlow server.

These links are at the top of the table:

- Click the **Settings** link to go to the Edit Server Settings page.
Use this page to edit properties for the ElectricFlow server.
- Click the **Access Control** link to go to the Access Control page for the server.
See the [Access Control](#) Help topic for more information.

The tables have this information:

- The System Access Control table displays top-level properties.
Click on an item in the Category column to see, add, or modify privileges for that object.
- In the Custom Server Properties table, click on a Property Name to view the property content.
If using *ectool*, these properties can be accessed using the property name starting with `"/server"`.

See the [Properties](#) Help topic for more information.

These links are provided within the table:

- **Create Property** - Use this link to create a new server property.
- **Create Nested Property** - Use this link to create a new nested server property.
- **Access Control** - This access control link allows you to create or edit privileges for the property sheet.
- The Actions column allows you to Edit or Delete the property in that row.

Settings - edit existing property settings

Use this page to edit property settings for the ElectricFlow server or any integrations you are using with ElectricFlow.

For example, if you access this page from the Server page, the page title will be Edit Server Settings and you will see fields for editing ElectricFlow server properties. From Perforce, the page will be titled Edit Perforce Settings, and the fields you can edit will be specific to Perforce functionality.

The following screen example is from the Edit Server Settings page.

Note: Depending on the ElectricFlow version you are using, the Edit Server Settings page may contain more or fewer fields than shown below.

For more information about any setting, hover your mouse over the setting name and click to see an inline Help description about the field value.

You may change any existing information or enter new information to empty fields. For Help information, hover your mouse over a field to see a Tool Tip.

Click **OK** when your changes are complete.

Also notice the "star" icon in the upper right corner. Click this icon to add this page to your Home page for quick access if you need to change these settings frequently.

Source Control Configurations

This web page displays all source control configurations you have created to communicate with ElectricFlow.

[Link at the top of the table](#)

- Use the **Create Configuration** link to create a new or additional source control configuration.

[Table column definitions](#)

Configuration Name - This is the name you chose for this source control configuration.

Description - This is the plain text or HTML description you provided for this source control configuration.

Plugin - This is the ElectricFlow plugin associated with this source control configuration.

Actions -

Edit - Use this link to modify information previously supplied to configure this source control system.

Delete - Use this link to remove this source control configuration.

Source Control Configurations - create new or edit existing configuration

Use this page to define a new source control configuration (SCM) or modify an existing source control configuration. A configuration is a collection of properties that define how to communicate with a particular source control system.

ElectricFlow bundles and supports numerous source control types.

- After creating a source control configuration, your entry will appear in the table on the Source Control Configurations web page— to see this web page, select the Administration > Source Control tabs.
- Other SCM systems are bundled with ElectricFlow and available in the Plugin Catalog. Each integration contains options specific to that SCM.
- The following SCM integrations are provided as examples.

[AccuRev](#)

[ClearCase](#)

[File](#)

[Git](#)

[Perforce](#)

[Property](#)

[Subversion](#)

Note: If you need a different SCM, not specified in this Help topic, go to the Plugin Manager page to locate your SCM (Administration > Plugins). See the Help topic associated with your plugin - located on the Plugins page. At a minimum, you need to enter a name for your configuration and you may want a minimal text description.

AccuRev

Enter information in the fields as follows:

Configuration Name - Type a name for the configuration— any name you choose, but the name must be unique. For example, you might use "myAccuRevServer."

Description - You can change or modify the default text description - ElectricFlow does not use this information.

Login As

User Name - The name ElectricFlow needs to use to communicate with your AccuRev system. For example, you may be using a special "read-only" user name similar to "Build" for your user name.

Password - The password for the specified User Name.

Retype Password - Type the password again.

Click **Submit** to save your information.

ClearCase

Enter information in the fields as follows:

Configuration Name - Type a name for the configuration— any name you choose, but the name must be unique. For example, you might use "myClearCaseServer."

Description - You can change or modify the default text description - ElectricFlow does not use this information.

Click **Submit** to save your information.

File

You may want to use this configuration with ElectricSentry to "watch" for changes in any files.

Enter information in the fields as follows:

Configuration Name - Type a name for the configuration—any name you choose, but the name must be unique.

Description - You can change or modify the default text description - ElectricFlow does not use this information.

Click **Submit** to save your information.

Git

Enter information in the fields as follows:

Configuration Name - Type a name for the configuration— any name you choose, but the name must be unique.

Description - You can change or modify the default text description - ElectricFlow does not use this information.

Click **Submit** to save your information.

Perforce

Enter information in the fields as follows:

Configuration Name - Type a name for the configuration— any name you choose, but the name must be unique. For example, you might use "myPerforceServer."

Description - You can change or modify the default text description - ElectricFlow does not use this information.

Login As

User Name (P4USER) - The name ElectricFlow needs to use to communicate with your Perforce system. For example, you may be using a special "read-only" user name similar to "Build" for your user name.

Password (P4PASSWORD) - The password for the specified User Name.

Retype Password - Type the password again.

P4PORT - Sets the hostname and port number ElectricSentry uses to contact the Perforce server (hostname:1234).

P4HOST - The name of the host machine to impersonate.

P4TICKETS - The full path name of the ticket file used by the "p4 login." This value is set as an environment variable, not as a command-line option (`fullPathAndFileName`)

P4CHARSET - Sets the character set used for translation of Unicode files (`characterSet`)

P4COMMANDCHARSET - Used to support UTF-16 and UTF-32 character sets (`commandCharSet`).

Click **Submit** to save your information.

Property

You may want to use this configuration with ElectricSentry to "watch" for any property changes.

Enter information in the fields as follows:

Configuration Name - Type a name for the configuration—any name you choose, but the name must be unique.

Description - You can change or modify the default text description - ElectricFlow does not use this information.

Click **Submit** to save your information.

Subversion

Enter information in the fields as follows:

Configuration Name - Type a name for the configuration— any name you choose, but the name must be unique. For example, you might use "mySubversionServer."

Description - You can change or modify the default text description - ElectricFlow does not use this information.

Login As

User Name - This is the name ElectricFlow needs to use to communicate with your Subversion system. For example, you may be using a special "read-only" user name similar to "Build" for your user name.

Password - This is the password for the specified User Name.

Retype Password - Type the password again.

Click **Submit** to save your information.

Checking out code from a source control system

Go to Projects > select a project > select a procedure.

To create a New Step for source code management, select the **Plugin** link.

- In the Choose Step panel, select Source Code Management from the left pane, then select the SCM you configured.
- The right-pane now shows the types of steps available for your configuration. Select the step you need and automatically go to the New Step page.

- On the New Step page, notice the Subprocedure section now contains the SCM integration you configured and the step you chose.
- Enter the remaining information to create your SCM step.

To edit an existing source control configuration

Modify any of your source control configuration information by "typing-over" any previously entered information.

Click **Submit** to save your modified source control information.

Active Users

This page displays all users known to this ElectricFlow server, including users defined locally within the server and those defined in external repositories such as LDAP or ActiveDirectory.

Active Users are those persons who have logged into ElectricFlow, although if a another user views a selection of user accounts, those users [viewed] will appear on the Active Users page also.

Click the **Show Inactive Users** link to go to the All Users web page—this page lists all users: local, LDAP or Active Directory, active, or inactive. If you delete an LDAP or Active Directory user from this [Active Users] page, the user will not be deleted from the All Users page.

- To edit **local** user content, click on the user name you want to modify.
- To create a new local user, click the **Create Local User** link at the top, right-side of the table.
- To delete a local user, click the **Delete** link in the Action column for that user.
- Click the "star" icon at the top, right-side of the table if you want to add this page to your Shortcut section on the Home page.

While you can view **external** user content, you **must** go to the LDAP or ActiveDirectory repository to edit external user information. However, you may associate properties with external users and then use those properties in ElectricFlow.

To configure your existing LDAP and Active Directory account repositories to communicate with ElectricFlow, click the Administration > Directory Providers tabs.

Users and Groups

Use this page to view a filtered list of ElectricFlow local users or groups.

Enter information in the fields as follows:

Field Name	Description
Filter	Enter a partial name to retrieve the list of all users or groups that match that name.

Field Name	Description
Maximum Results	Enter the number of results you would like to see returned. Default is 100. Caution: Currently, ElectricFlow has no upward limit, but if you enter a number of 1000 or higher, you may overload your browser, which will result in performance degradation or worse.
Include Inactive [User/Group]	Use this check box only if requesting external LDAP or Active Directory providers. <ul style="list-style-type: none"> • If "checked," all available users or groups are returned. • If not checked, only those users or groups known to the ElectricFlow server are returned, for example, users who have logged in or groups containing users who have logged in.

Click **OK** to begin the search.

[Links at the top of the page](#)

Create Local [User/Group] - Click this link to create a new user or group - for users or groups *local* to the ElectricFlow server.

[Column descriptions](#)

Column Name	Description
Name	The name of the user or group. The name combined with the location is the unique identifier for this user or group, <i>local</i> to the ElectricFlow server. Click the name to see more information.
Repository	The repository where the user or group is located. "Local" means local to the ElectricFlow server. Other names in this column refer to defined directory providers.
Location	Only viewable for users or groups from non-local directory providers.
Real Name (users only)	The full name of the user.
Email (users only)	The email address for the user.

Column Name	Description
Actions	<p>Edit - Only available for local users and groups. Click to edit the user or group definition.</p> <p>Delete - Click to delete the user from the ElectricFlow server.</p> <p>Note: This delete function does not delete non-local users from an LDAP or Active Directory repository.</p>

User - create new or edit existing local user

Use this page to create or modify a *local* user **only**.

ElectricFlow supports two kinds of user accounts: those defined externally in an LDAP or Active Directory system, and *local* users defined in this ElectricFlow server.

Local users are not visible outside ElectricFlow.

Electric Cloud recommends using external accounts whenever available, but you may need to create local users if you do not have a shared directory service or if you need special accounts to use for ElectricFlow only.

To create a new local user

Enter information in the fields as follows:

Field Name	Description
User Name	Enter the name of the user account to be used for login
Real Name	Enter the user's real (given) name.
Email	Enter the user's email address.
Password	Enter the user's login password.

Click **OK** to save the new local user data.

To edit an existing local user

- You may highlight and type-over any previously entered user information.
- If you leave the password fields blank and click **OK**, the existing password remains unchanged.
- To change the user's password, enter the new password for the user, AND also enter the current admin password.

Click **OK** to save the modified user data.

Use this page to modify user properties or assign Custom User Properties. Select the **Create Property**, **Create Nested Sheet**, or **Access Control** links to set the properties you need.

Note: To configure your existing LDAP and Active Directory account repositories to communicate with ElectricFlow click the **Administration > Directory Providers** tabs.

User Details

This page displays external user information retrieved from a repository such as LDAP or Active Directory.

To edit external user information, you **must** connect directly to the repository to modify external user content. However, you can associate properties with an external user and then use those properties in ElectricFlow.

Use this page to modify user properties or assign Custom User Properties. Select the **Create Property**, **Create Nested Sheet**, or **Access Control** links to set the properties you need.

Definitions for summary information at the top of page

User Name: The name this user is known by in the system.

Repository: Usually LDAP or Active Directory.

Real Name: The real, given name for user.

Email: The user's email address as known within the system.

Groups: This field displays all groups where this user is a member. Clicking on a group name displays the Group Details page, which lists all members who belong to that group.

Note: To configure your existing LDAP and Active Directory account repositories to communicate with ElectricFlow, click the Administration > Directory Providers tabs.

Edit User Settings

Use this page to modify your User Settings for the product tab view. Depending on which view you choose, you can have limited access to ElectricFlow features and functions.

In the Tab View, use the pull-down menu to make a new selection:

Default - Selecting this view provides all regular/standard product tab views, including plugins, which by default are installed into this view.

Base - This view removes the ability to use a plugin. You can see the plugin list (accessed by selecting the Plugins tab), but you are denied access to configure, install, uninstall, or promote a plugin.

Inherit - Frequently this view is set for a group. The UI searches for the property, `ec_ui/defaultView`, first on the user, second in any groups the user belongs to, and last, on the server. The inherit setting clears the user's private defaultView setting and then picks up the first value from the rest of the search path.

Click **OK** after selecting a different tab view.

Groups

This page displays all groups known to this ElectricFlow server, including groups defined locally within the server and those groups defined in external repositories such as LDAP.

- To edit a **local** group, click on its name.
- To create a new local group, click the **Create Local Group** link.
- Click on an **external** group to view its contents, but you cannot edit the content through ElectricFlow. Instead, you must go to the group's repository directly. You can, however, associate properties with external groups, which can then be used in ElectricFlow.

To configure your existing LDAP and Active Directory account repositories to communicate with ElectricFlow, click the Administration>Directory Providers tabs.

Group - create new or edit existing local group

To create a new local group

Enter information in the fields as follows:

Name - Type a name for the group. Choose any name, but it must be unique among other local group names.

Users - Type a list of users who need to belong to this group— type one user per line.

Click **OK** after populating the group.

To edit a local group

- You can highlight the existing group name and type a new name—or
- You can scroll to the bottom of the list of users and type-in additional members for this group, one name per line—or
- You can highlight a user name and press the delete key (on your keyboard) to delete that user from the group.
- Click **OK** after completing your changes.

Also, you can associate custom properties with a group. Select either the **Create Property**, **Create Nested Sheet**, or the **Access Control** links to set up the properties you need.

Group Details

This page displays external group information retrieved from a repository such as LDAP or ActiveDirectory.

To edit external group information, you must connect directly to the repository to modify an external group. However, you can associate properties with an external group and then use those properties in ElectricFlow.

Click the **Access Control** link [at the top of the table] to see the privileges assigned to this group, inherited privileges, add a user or group, and so on.

Also, you can associate custom properties with a group. Select either the **Create Property**, **Create Nested Sheet**, or the **Access Control** links to set up the properties you need.

Note: To configure your existing LDAP and Active Directory account repositories to communicate with ElectricFlow, click the Administration > Directory Providers tabs.

Directory Providers

ElectricFlow uses account information from multiple sources. In most cases, the primary account information source is an external LDAP or Active Directory repository: both user and group information is retrieved from the repository.

Local users and groups are defined within ElectricFlow.

Links at the top of the table

Click the **Add Active Directory Provider** or the **Add LDAP Provider** link to go to a new web page to configure your Directory Provider.

After configuring your directory provider, that name will appear in the All Providers table, the Provider Name column.

Column descriptions

Column Name	Description / Actions
Provider Name	Click on a Provider Name to make changes to an existing directory provider.
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code>
Actions	<p>Copy - Use this link to make a copy of the Provider Name configured on this row.</p> <p>Remove - Use this link to remove the Provider Name on this row.</p>

Detailed Help for defining a directory provider in ElectricFlow is available from the [New Active Directory Provider](#) or the [New LDAP Provider](#) web page.

Click the **Help** link from either web page. The Help text also contains examples you may find useful.

Directory Providers - create new or edit existing directory providers

ElectricFlow supports Active Directory and LDAP directory providers

This Help topic contains instructions for creating an Active Directory or LDAP directory provider and editing those configurations if you need to make changes at a later time. Use the following links to go the section you need:

[To create a new Active Directory provider](#)

[To create a new LDAP directory provider](#)

[Examples for directory provider field descriptions](#)

[To edit an existing directory provider](#)

To create a new Active Directory provider

Enter information in the fields as follows to specify your existing Active Directory users and groups to communicate with ElectricFlow. For examples of information you enter in the these fields, see the [table](#) after the following description sections.

Field Name	Description
<i>General section</i>	
Provider Name	This name identifies users and groups that come from this provider.
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code>
URL Discovery	Select the method to retrieve the URL for the Active Directory server. Auto-discovery using DNS automatically discovers Active Directory servers on the given domain. Alternatively, you can specify a custom URL to an Active Directory server.
Domain Name	The domain where Active Directory servers are automatically discovered. For example, <code>my-company.com</code>

Field Name	Description
Use SSL	<p>Select this box if you want to use SSL when the ElectricFlow server contacts your Active Directory server.</p> <p>Note: Transport Layer Security (TLS) has replaced Secure Sockets Layer version 3.0 (SSLv3) on the ElectricFlow web server and the ElectricFlow server.</p>
URL	<p>This is an explicit URL to the Active Directory server. The URL is in the form <i>protocol://host:port/basedn</i>. Protocol is either <i>ldap</i> or <i>ldaps</i> (for secure LDAP). The port is implied by the protocol, but you can override the port if you cannot use the default location (default is port 389 for <i>ldap</i> or 636 for <i>ldaps</i>). The <i>basedn</i> is the path to the top-level directory that contains users and groups at this site. Typically, this is the domain name where each part is listed with a <i>dc=</i> and separated by commas. Note: Spaces in the <i>basedn</i> must be URL encoded (use <i>%20</i> for spaces). If your site uses redundant directory servers for failover, you can enter multiple URLs separated by spaces.</p>
Query User Name	<p>The name of a user who has read-only access to user and group directories in Active Directory. This is the user name to use for fetching user and group information. When you provide a domain name, you can provide the simple name, for example, <i>myuser</i>. When you provide an explicit URL, you need to provide a distinguished name, for example:</p> <p><i>cn=myuser,ou=Users,dc=electric-cloud,dc=com.</i></p>
Query User Password	The password for the query user.
<p>User Options section</p> <p>Note: When creating an Active Directory provider, the ElectricFlow server automatically sets default values for any options (fields) that are empty. The default values match the most common Active Directory configurations. After the provider is created, you can view and modify defaults by modifying the provider.</p>	
User Base	This string is prepended to the <i>basedn</i> to construct the directory DN containing user records.
User Search Filter	This LDAP query is performed in the context of the user directory to search for a user by account name. The string " <i>{0}</i> " is replaced with the user's login ID. Typically, the query compares a user record attribute to the substituted user login ID.
User Name Attribute	This is the attribute in a user record that contains the user's account name.

Field Name	Description
Full User Name Attribute	(optional) This is the attribute in a user record that contains the user's full name (first and last) for display in the UI. If this attribute is not specified or the resulting value is empty, the user's account name is used instead.
User Email Attribute	(optional) This is an attribute in a user record that contains the user's email address. If this attribute is not specified, the account name and domain name are concatenated to form an email address.
Search User Subtree	Select this check box to search the specified directory by the user base and all directories below. If this check box is not selected, the search is limited to the specified directory only.
<p>Group Options section</p> <p>Note: If you do not need group information at this time, do not enter information in the Group section, the group test fails without this information, but the test successfully authenticates users if both "user tests" are successful.</p> <p>When creating an Active Directory provider, the ElectricFlow server automatically sets default values for the options/fields that remain empty. These default values match the most common Active Directory configurations. After the provider is created, you can view and modify the defaults by modifying the provider.</p>	
Enable Groups	Select this check box to enable external groups for this directory provider.
Group Base	(optional) This string is prepended to the <code>basedn</code> to construct the directory DN containing group records.
Group Member Filter	(optional) This LDAP query is performed in the groups directory context to identify groups that contain a specific user. Two common forms of group records in LDAP directories are: <code>POSIX</code> style groups where members are identified by account name, and <code>groupOfNames</code> or <code>uniqueGroupOfNames</code> records where members are identified by the full user DN. ElectricFlow supports both forms, so the query is passed two parameters: " <code>{0}</code> " is replaced with the full user record DN, and " <code>{1}</code> " is replaced with the user's account name.
Group Member Attributes	(optional) This is a comma-separated attribute name list that identifies a group member. Most LDAP configurations only specify a single value, but if a mixture of <code>POSIX</code> and <code>LDAP</code> style groups exist in the directory, multiple attributes might be required.

Field Name	Description
Group Search Filter	(optional) This LDAP query is performed in the context of the groups directory to enumerate group records. You can choose from common templates that include either security or distribution groups (or both). These templates are based on the most common Active Directory settings.
Unique Group Name Attribute	(optional) This is the group record attribute that contains the group name. To prevent group name overlap between multiple directory providers (or within the same provider in a multi-forested Active Directory server), Electric Cloud recommends setting this attribute to the <code>distinguishedName</code> .
Common Group Name Attribute	The Unique Group Name Attribute may not be searchable if using the <code>distinguishedName</code> . In this case, the Common Group Name Attribute can be used to search for groups, depending on the filter used.

After filling-in all fields, click the **Test** button.

Three tests validate the information you supplied:

- user authentication
- user identified in Active Directory
- find all groups where the user is a member

If there is a test failure, correct the information you supplied and retest.

Click **Save** after successful test results.

New, defined directory providers will appear in the table on the Directory Provider web page.

To create a new LDAP directory provider

Enter information in the fields as follows to specify your existing LDAP users and groups to communicate with ElectricFlow. For examples of information you enter in the these fields, see the table after the following description sections.

Field Name	Description
General section	
Provider Name	This name identifies users and groups that come from this provider.

Field Name	Description
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code>
URL	The LDAP server URL is in the form <i>protocol://host:port/basedn</i> . Protocol is either <i>ldap</i> or <i>ldaps</i> (for secure LDAP). The port is implied by the protocol, but you can override the port if you cannot use the default (default port is 389 for <i>ldap</i> and 636 for <i>ldaps</i>). The <i>basedn</i> is the path to the top-level directory that contains users and groups at this site. Typically, this is the domain name where each part is listed with a <i>dc=</i> and separated by commas. Note: Spaces in the <i>basedn</i> must be URL encoded (use <i>%20</i> for spaces). If your site uses redundant directory servers for failover, you can enter multiple URLs separated by spaces.
Realm	This is the realm of the LDAP directory provider, which is used to create unique user names when you have multiple providers. For example, if the realm is <i>my-company.com</i> , users are saved as <i>myuser@my-company.com</i> .
Query User Name	This is the name of a user who has read-only access to the user and group directories in LDAP. This is the user name to use for fetching user and group information. When providing a domain name, you can provide the simple name, for example, <i>myuser</i> . When providing an explicit URL, you need to provide a distinguished name, for example: <i>cn=myuser,ou=Users,dc=electric-cloud,dc=com</i> .
Query User Password	This is the password for the query user.
User Options section	
User Base	This string is prepended to the <i>basedn</i> to construct the directory DN containing user records.
User Search Filter	This LDAP query is performed in the context of the user directory to search for a user by account name. The string " <i>{0}</i> " is replaced with the user's login ID. Typically, the query compares a user record attribute to the substituted user login ID.
User Name Attribute	This is the attribute in a user record that contains the user's account name.

Field Name	Description
Full User Name Attribute	(optional) This is the attribute in a user record that contains the user's full name (first and last) for display in the UI. If this attribute is not specified or the resulting value is empty, the user's account name is used.
User Email Attribute	(optional) This is the attribute in a user record that contains the user's email address. If the attribute is not specified, the account name and domain name are concatenated to form an email address.
Search User Subtree	Select this check box to search the specified directory by the user base and all directories below. If this check box is not selected, the search is limited to the specified directory only.
<p>Groups Options section</p> <p>Note: If you do not need group information at this time, do not enter information in the Group section, the group test fails without this information, but the test successfully authenticates users if both user tests are successful.</p>	
Enable Groups	Select this checkbox to enable groups.
Group Base	(optional) This string is prepended to the <code>basedn</code> to construct the directory DN containing group records.
Group Member Filter	(optional) This LDAP query is performed in the groups directory context to identify groups containing a specific user. Two common forms of group records in LDAP directories are: <code>POSIX</code> style groups where members are identified by account name, and <code>groupOfNames</code> or <code>uniqueGroupOfNames</code> records where members are identified by the full user DN. ElectricFlow supports both forms, so the query is passed with two parameters: " <code>{0}</code> " is replaced with the full user record DN, and " <code>{1}</code> " is replaced with the user's account name.
Group Member Attributes	(optional) This is a comma-separated attribute name list identifying a group member. Most LDAP configurations only specify a single value, but if you have a mixture of <code>POSIX</code> and <code>LDAP</code> style groups in the directory, multiple attributes might be required.
Group Search Filter	(optional) This LDAP query is performed in the context of the groups directory to enumerate group records.
Unique Group Name Attribute	(optional) This is the group record attribute containing the group name.

Field Name	Description
Common Group Name Attribute	The Unique Group Name Attribute may not be searchable if using distinguishedName. In this case, the Common Group Name Attribute is used if searching for groups, depending on the filter.

After filling-in all fields, click the **Test** button.

Three tests validate the information you supplied:

- user authentication
- user identified in LDAP
- find all groups where the user is a member

If there is a test failure, correct the information you supplied and retest.

Click **Save** after successful test results.

New, defined directory providers will appear in the table on the Directory Provider web page.

Examples for directory provider field descriptions

The following table provides examples for filling in the fields described above:

Field Name	LDAP example	ActiveDirectory example
Provider Type	LDAP	ActiveDirectory
Domain Name	N/A	my-company.com
Realm	my-company.com	N/A
URL	ldap://dir.company.com/dc=company,dc=com	ldaps://server/dc=company,dc=com
Query User Name	uid=JohnDoe,ou=People,dc=company,dc=com	cn=myuser,cn=Users,dc=company,dc=com

Field Name	LDAP example	ActiveDirectory example
Query User Password	mypw	mypw
User Base	ou=People	cn=Users
User Search Filter	uid={0}	(&((userPrincipalName={0}) (sAMAccountName={0})) (objectClass=user))
User Name Attribute	uid	userPrincipalName
Full User Name Attribute	gecos	name
User Email Attribute	mail	mail
Group Base	ou=Group	cn=Groups
Group Member Filter	((member={0})(memberUid={1}))	member={0}
Group Member Attribute	member,memberUid	member

Field Name	LDAP example	ActiveDirectory example
Group Search Filter	<code>((objectClass=groupOfNames) (objectClass=posixGroup))</code>	<code>(objectClass=group)</code>
Unique Group Name Attribute	<code>distinguishedName</code>	<code>distinguishedName</code>
Common Group Name Attribute	<code>cn</code>	<code>cn</code>

To edit an existing directory provider

You may change any previously supplied information in the fields.

After editing any fields, click the **Test** button.

The same three tests validate the information you supplied:

- user authentication
- user identified in the directory provider
- find all groups where the user is a member

If there is a test failure, correct the information you supplied and retest.

Click **Save** after successful test results.

Edited, redefined directory providers will appear in the table on the Directory Provider web page.

Test Directory Provider

You must enter a valid User Name and Password on this page. Omitting or using an incorrect User Name and/or Password results in the test not completing or completing with an error. In either case, you will not obtain successful test results even if the Directory Provider information you supplied is valid and correctly entered into ElectricFlow.

Three tests validate the information you supplied:

- user authentication
- user identified in a directory provider

- find all groups where the user is a member

If there is a test failure, correct the information you supplied and retest by clicking the **Test** button.

If you do not need group information at this time, and did not enter information in the Group section, the group test fails without this information, but the test successfully authenticates users if both user tests are successful.

Workflows

This page displays all workflows in the ElectricFlow system, both running and completed workflows.

Links and actions at the top of the table

- Sort the Name, Project, State, Modify Time, Workflow Definition, and Completed columns by clicking on the column name.
After sorting a column, the page changes to a Workflow Search Results page and more search options are available.
- Define a search to see only the workflows you choose. Click the **New Search** link.
- The "star" icon allows you to save this web page to your Home page.

Column descriptions

Name - This is the ElectricFlow-generated workflow name, which is defined by the Workflow Name Template setting. Click on any workflow name to go to that workflow's Workflow Details page.

Project - The name of the project containing this workflow. Click on any project name to go to the Project Details page for that project.

State - The current state of the workflow.

Modify Time - The time this workflow was modified.

Workflow Definition - The name of the workflow definition that ran to create this workflow. Click on any workflow definition name to go to that workflow definition's Workflow Definition Details page.

Completed - A "checkmark" in this column indicates the workflow has completed.

Actions - **Delete** - Use this link to delete the workflow on the same row.

Workflow Definition - create new or edit existing workflow definition

To create a new workflow definition

Enter information in the fields as follows:

Field Name	Description
Name	Enter a unique workflow definition name of your choice. You may want your workflow definition name to reflect the project where it belongs or its purpose. For example, you might set a name based on a product or team using this workflow.
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code>
Workflow Name Template	<p>This is the template used to determine the default name of jobs launched from the workflow definition.</p> <p>For example:</p> <pre> \$[projectName]_\${/increment /myproject/workflowCounter}_ \${/timestamp} </pre> <p>(substitute your values for the names above)</p> <p>Produces a workflow name like:</p> <pre> projectFoo_123_20140102130321 </pre> <p>Enter any combination of elements to create workflow names more meaningful to you. For example, you could choose to include the build number.</p> <p>Note: Electric Cloud does not recommend using the ElectricFlow-generated <code>workflowId</code> because it is no longer a human-readable integer so it does not provide any identifiable information and cannot be used as a counter.</p>

Click **OK** to save the information you entered.

Your new workflow definition will appear on the Project Details > Workflow Definitions page.

To edit an existing workflow definition

1. To rename the workflow definition, type a new name in the **Name** field and click **OK**.
2. Change or add to the description or modify the template.
3. Click **OK** when your edits are complete.

Workflow Definition Details

This Help topic provides a detailed overview for all views and functionality on this web page. You may want to read this topic in its entirety if this is your first experience with the ElectricFlow Workflow feature. If you are an experienced workflow user, the following quick links will take you to a specific topic section for quick review.

A note to existing workflow users: We recommend reviewing the Graph view section because workflow functionality previously found on other ElectricFlow web pages is now combined for ease-of-use on the Graph page.

[Graph view](#)

[To create a New State](#)

[To create a State Definition](#)

[Action](#)

[Notifiers](#)

[Parameters](#)

[Properties](#)

[To create a Transition Definition](#)

[Using the graph's quick-access features](#)

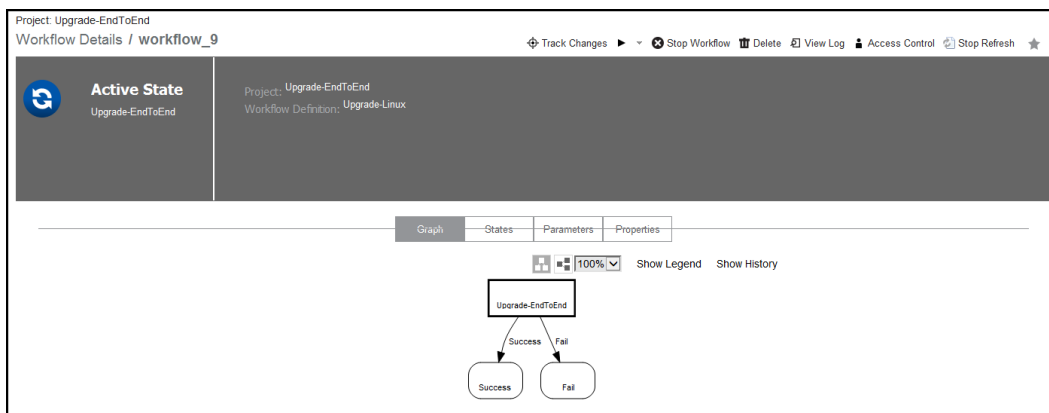
[Show Legend](#)

[List view](#)

[Properties view](#)

This page displays workflow definition components, including state and transition definition names, types, actions, descriptions, and various links to create additional transitions, state definitions, define access control, run the workflow, and so on. You can view any previously defined workflow objects and create new properties too.

Opening to the Graph view, your first glance at this page will look similar to the following:



Links and actions at the top of the page:

- Above the Workflow Definition Details page title you see (in this example) "Project: Upgrade-End to End". This breadcrumb information tells you which project you are viewing or working with and provides a link back to the project, itself.
- Immediately after the page title, you see "workflow_9", which (for our example) is the name of this workflow.



- Track Changes ()—Click this to open the Change History page.



- **Run** ()- Use this link to run the workflow definition. Hover your mouse over the drop-down arrow to see these choices:
 - Selecting **Run...** - allows you to pick the "starting state" to run the workflow. After clicking **OK** from the Run Workflow pop-up dialog box, the Run Workflow page is displayed. If you previously created parameters for this workflow, they are displayed. You can accept the parameter values or change them before running the workflow.
 - Selecting **Run Immediately** - this option uses the first (default) "starting state" to run the workflow.
Note: Clicking Run, without using the down-arrow to make a selection, is the same as selecting Run Immediately.
- **Access Control** - Use this link to set privileges for this workflow definition. ***For more information***, see the Access Control Help topic.
- The "star" icon allows you to save this workflow definition to your Home page.

Navigation and view summary

Each button has its own expanded description section following this summary.

- Graph - the page opens in this view. The main portion of this view contains the work area for visually creating your workflow.
- List - this view provides a table displaying states, transitions, and so on in list form.
- Properties - this view provides a table displaying previously created properties and provides links to create additional properties.
- Show Legend - displays a legend for workflow objects.
- Create State - this button produces the State Definition panel where you can create new states as you build or modify your workflow.
- The multi-square directional icons allow you to change your workflow graph from vertical to horizontal, depending on how you choose to see the graph.
- The drop-down menu for percentages allows you to diminish or expand the workflow graph size.

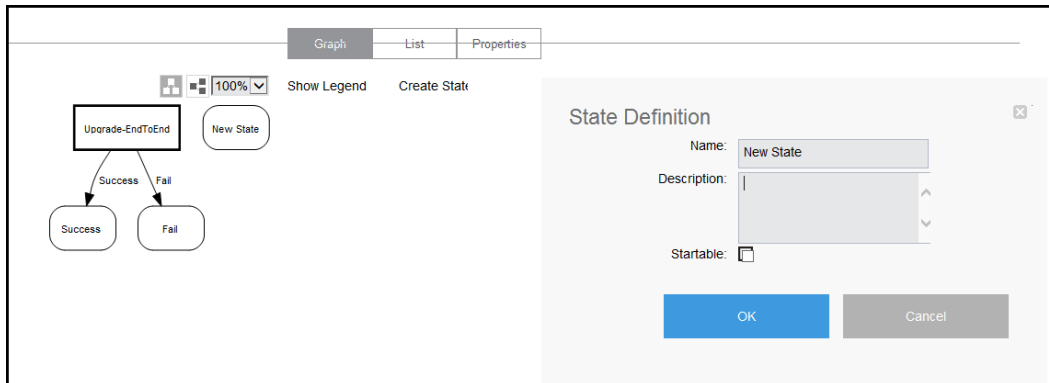
Graph view

To create a workflow, minimally you need to create state definitions and transition definitions. Optimally, those objects will have parameters and properties, and you will want to create or enforce access control and perhaps

create one or more email notifiers too. To begin, the following sections describing functionality in this view will familiarize you with how to create workflow objects and end with exploring the graph's quick-access features.

Click the **Create State** button to create a new state using the State Definition panel.

In the following screen example, you see a New State object added next to the Start state in an existing workflow graph.



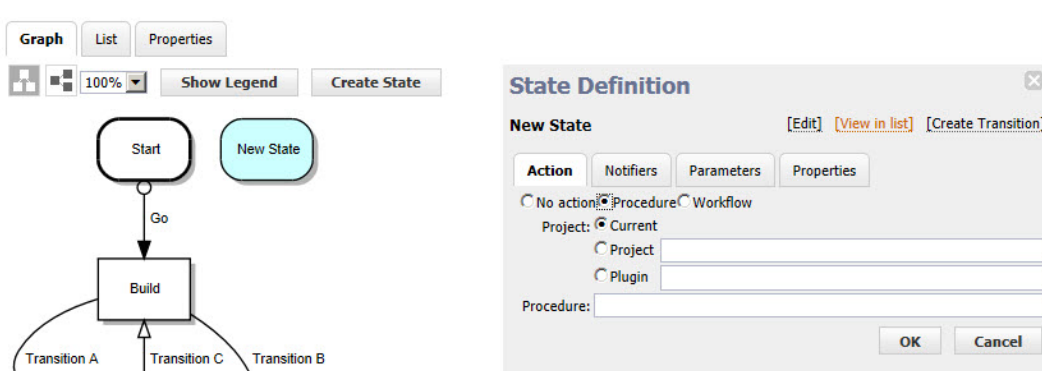
To create a new state

After clicking the Create State button, enter information in the State Definition panel as follows:

Field Name	Description
Name	(Required) Provide any name of your choice. The name must be unique to this workflow definition. If you do not provide a name, a system-generated name is created: New State, New State 2, and so on.
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code> .
Startable	Select this checkbox if you want this state to be your workflow "starting" state. The first state in any workflow definition is saved as <i>Startable</i> , but you can change this later.

Click **OK** and the State Definition panel provides options to configure your new State Definition.

To configure the new State Definition



In the State Definition Panel, the state name appears at the top. In this example in this topic, "New State" is the state name.

Configure the following settings.

Links at the top of the panel

- **Access Control** - use this link to add or modify access control privileges for this state (link not illustrated in the State Definition panel example above).
- **Edit** - use this link to modify this state's name, description, or whether or not the state is *Startable*.
- **View in list** - displays the List view to see the state definition in the Workflow Definition Details table. This state will be highlighted in the List view.
- **Create Transition** - displays the Transition Definition panel to create a transition for this state.

The "tabbed" or button sections

Action

As each state in a workflow becomes active, it performs an *action*. The state's action can be to create a job from a procedure or to start a workflow from a workflow definition. When the job or called workflow completes, On Completion transitions are evaluated to see if the workflow should change to a different state, possibly based on the outcome of the action.

Selecting *No action*, *Procedure*, or *Workflow* changes the fields that appear under these choices.

- **No action** - if selected, no other information is required. Click **OK**.
- **Procedure** - this selection displays the following fields:
 - **Current** - this selection refers to the project that contains the workflow definition.
 - **Project** - if selected to call a different project, start typing in the text box to display a list of (non-plugin) projects from which to select. Only the first 10 matches are displayed. Type more characters to refine the list.
 - **Plugin** - if selected, start typing in the text box to display a list of plugin projects from which to choose. Only the first 10 matches are displayed. Type more characters to refine the list.

- Procedure - start typing in this field to see a list of available procedures from which to select, depending on your previous choice of Current, Project, or Plugin. Only the first 10 matches are displayed. Type more characters to refine the list.
- Workflow - this selection displays the following fields:
 - Similar to selecting Procedure (above), select Current, Project, or Plugin.
 - Workflow - if selected to call another workflow definition, start typing in this text box to display a list of workflow definitions from which to choose. Only the first 10 matches are displayed. Type more characters to refine the list.
 - Starting State - start typing in this field to see a list of available starting states definitions in the workflow definition you selected. Only the first 10 matches are displayed. Type more characters to refine the list.

After selecting the procedure or starting state, press the <Tab> key to leave the field and see a list of parameters for the selected action. After entering any procedure values, click **OK** to save the changes.

Notifiers

Select this tab to see a list of email notifiers previously created for this state definition or to create an email notifier. Notifiers are commonly used to inform interested parties about transitions to the state being defined.

Note: Before you can set up an email notifier, you need an email configuration. If you have not already defined an email configuration, you can do it now. Go to Administration > Email Configurations and select the **Add Configuration** link.

The Notifiers table contains a list of all previous created notifiers for this state definition.

Column descriptions

Column Name	Description / Actions
Name	The name of the email notifier. Select a name to go to the "edit" page for that notifier if you need to make changes.
Type	The type of email notifier specified during creation.
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code>
Actions	Delete - Use this link to delete the notifier on that row.

State Definition [X]

New State [Edit] [View in list] [Create Transition]

Action **Notifiers** Parameters Properties

Name	Type	Description	Actions
New Email Notifier	onEnter		Create Notifier Delete

To create an email notifier, click the **Create Notifier** link to access the Email Notifier panel.

Email Notifier [X]

Name: Required

Description:

Type:

Condition:

Template:

Subject: Required

Message body goes here.

Email Config:

Destination: Required

Enter information in the fields as follows:

Field Name	Description
Name	(Required) This name can be any text string you choose. The name must be unique among other notifier names in this project, on procedures or workflow definitions called from other projects.
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code>
Type	<p>Use the pull-down menu to choose a type:</p> <ul style="list-style-type: none"> On Completion Notifier - sends an email after the state's job or workflow completes. If no job or workflow is defined for that state, these notifiers will not be sent. On Enter Notifier - sends an email when the state becomes the workflow's active state. On Start Notifier - sends an email after the state's job or workflow starts. If no job or workflow is defined for that state, these notifiers will not be sent.
Condition	Use the pull-down menu to select the type of condition you need for this email notifier. Edit the auto-supplied condition in the text box or add a completely new script for your purpose. The condition specifies whether the notifier should send a message depending on the result of a property expansion. If the result is empty or non-zero, the message is sent. If the result is "0", the message is not sent.
Template	<p>Use the pull-down menu to select from a list of global, ready-to-use formatting templates. Depending on the type of email notifier you are creating, the available template choices in the drop-down menu will be different.</p> <p>(Required) To customize your template, edit the auto-supplied text in the text box or you can add a completely new script for your purpose. Any edits made in this text box will not be saved to the global template.</p> <p>To create a custom template, the basic structure is:</p> <ul style="list-style-type: none"> zero or more email header lines blank line message body

Field Name	Description
Email Configuration	Click inside this field or start typing to bring up a list of possible email configuration names. An email notifier that does not specify an email configuration will use the configuration named 'default' if it exists.
Destinations	(Required) This is a space-separated list of valid email addresses, email aliases, or ElectricFlow user or group names, or a property reference that expands into such a list, or you can enter an LDAP DL name (group name).

Click **OK** to save your email notifier configuration. The next time you view the Notifier table, you will see this notifier in the list.

Parameters

Similar to Notifiers, a parameter table is displayed containing all formal parameters defined for this state definition. Parameters are presented when launching the workflow definition (if it is startable), or when taking a manual transition to the state. A transition may enter values for some or all of the formal parameters defined by the state definition, in which case only the unmapped parameters are presented to the user.

- **Create Parameter** - Use this link to go to the New Parameter page to create a parameter for this state definition.

If you need assistance with creating a parameter, click the **Help** link in the upper-right corner of the New Parameter page.

Properties

Similar to the Notifiers and Parameters table, a properties table is displayed containing available properties for this state definition. If no properties were defined for this state definition, the table does not exist.

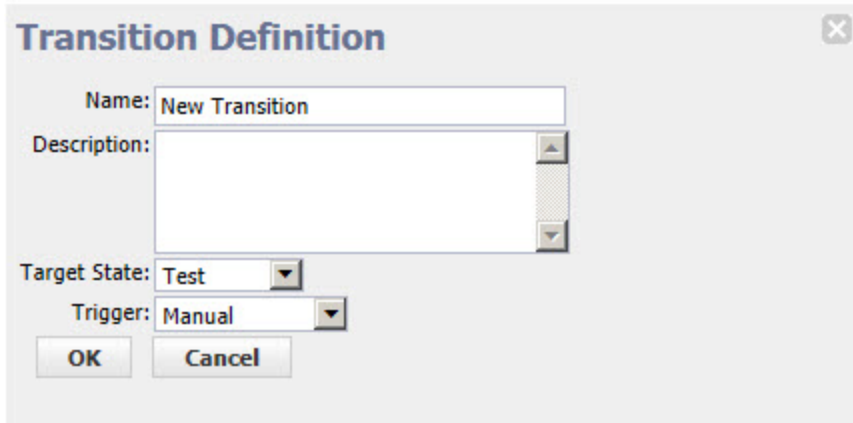
Click one of the available links to create one or more properties that will be displayed in the table when created:

- **Create Property** - Click this link to go to the New Property pop-up box to create a new property for this state definition.
- **Create Nested Sheet** - Click this link to go the New Nested Property Sheet pop-up box to create a nested property.
- **Access Control** - Click this link to go to the Access Control page for the property sheet.

The property pop-up boxes contain a Help link if you need assistance creating a property.

To create a Transition Definition

On the State Definitions panel, click the **Create Transition** link to see the Transition Definition panel:



The image shows a 'Transition Definition' dialog box. It has a title bar with a close button (X). Inside, there are four input fields: 'Name' with the text 'New Transition', 'Description' which is empty, 'Target State' with a dropdown menu showing 'Test', and 'Trigger' with a dropdown menu showing 'Manual'. At the bottom are 'OK' and 'Cancel' buttons.

Enter information in the fields as follows:

Field Name	Description
Name	(Required) Enter a unique name for the transition definition. It can be any name you choose, but the name must be unique within the state definition. If you do not provide a name, a system-generated name is created: <i>New Transition</i> , <i>New Transition 2</i> , and so on.
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code>
Target State	This field displays the "best guess" for the target state. If not correct, or you intended to create a transition for a different state, use the down-arrow to choose an available target state from the list.

Field Name	Description
Trigger	<p>Use the drop-down menu to choose the transition type.</p> <p>The four types of transitions are:</p> <ul style="list-style-type: none"> On Completion - transition occurs when the action completes. These transitions are ignored if no action is specified for the source state. Note: On Completion transitions are taken only if the state is still active when the action completes, and are ignored if the workflow has transitioned to another state—this can occur if an On Start or Manual transition occurred before the action completed. On Enter - transition occurs before sending notifiers or starting the action. On Start - transition occurs immediately after starting the action. These transitions are ignored if no action is specified for the source state. Manual - transition occurs when a user selects the transition in the UI and specifies parameters. The same action can occur using ectool or the Perl API by calling <code>transitionWorkflow</code>. Only users who have "execute" permission on the transition are allowed to use a Manual transition.
Condition	<p>Use the drop-down menu to choose a condition. After selection, the text box is populated with a sample JavaScript string to edit for your purposes.</p> <p>Note: If you select Custom, no sample is provided. Enter a script in the text box. We recommend using JavaScript.</p>

Click **OK** to save your transition definition.

Notice the new links and options available now:

- **Access Control**—Use this link to add or modify access control privileges to this transition definition.
- **Edit**—Use this link to modify the transition definition.
- **View in list**—Use this link to see the new transition definition in the List view (table) on the Workflow Definition Details page. You can use the List view to move the transition to a different position within the state definition.
- **Parameters**—If parameters were defined on the State Definition Details page, you will not see them here. If parameters are defined on this page, you need to enter values. Manual transitions allow you to defer parameter assignment until transition time.
- **Properties**—Displays properties available for this object or you can create one or more properties at this time.

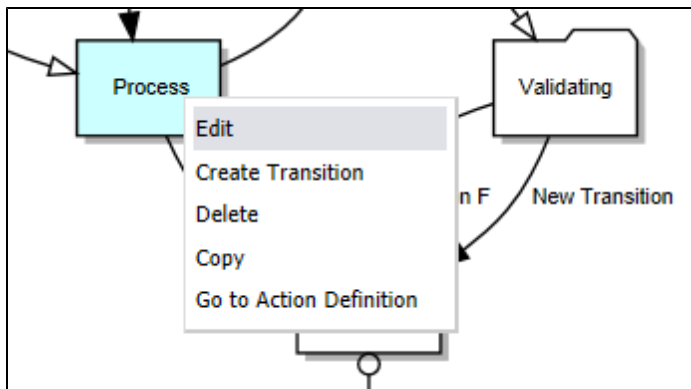
Using the graph's "quick-access" features

The following screen example is a portion of a workflow graph.

In the graph, states and transitions are links:

- Click on a state to open the State Definition panel to modify that state.
- Click on a transition to open the Transition Definition panel to modify that transition.
(Hover your mouse over a transition or transition name and click when it changes color.)

However, if you **right-click** on a state or transition, a context-sensitive menu with other options is available. In the following example, right-clicking on the "Process" state provided the quick-link pop-up menu.



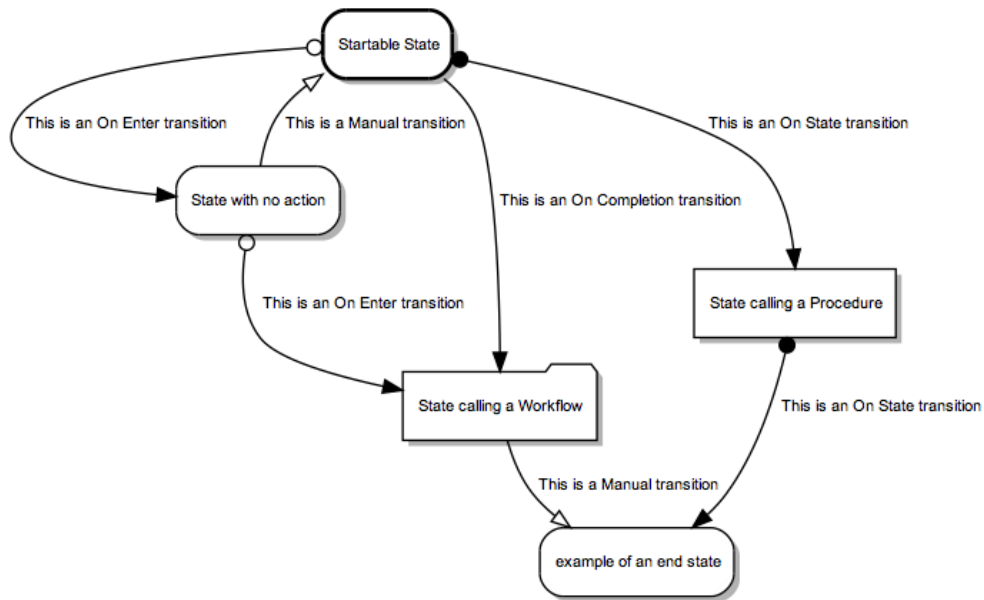
Available pop-up links include:

- For states calling a procedure or another workflow, menu choices can be:
 - Edit - opens the State Definition panel to make modifications. This is the same as selecting the Edit link on the State Definition panel.
 - Create Transition - opens the Transition Definition panel.
 - Delete - deletes the state.
 - Copy - makes a copy of the state with all associated transitions preserved.
 - Go to Action Definition - for states calling a procedure, the Procedure Details page is opened. For states calling a workflow, the Workflow Definitions Details page for the "subworkflow" is opened.
- For states with "No action", the choices are the same, except the "Go to Action Definition" option is not available.
- For transitions, menu options include: Rename, Delete, and Copy.
- Right-clicking anywhere on the graph canvas provides two more choices:
 - Create State - allows you to quickly create a new state. This is the same as clicking the Create State button at the top of the graph.
 - Rotate Graph - acts as a "toggle", rotating the graph from a vertical to horizontal view and back again.

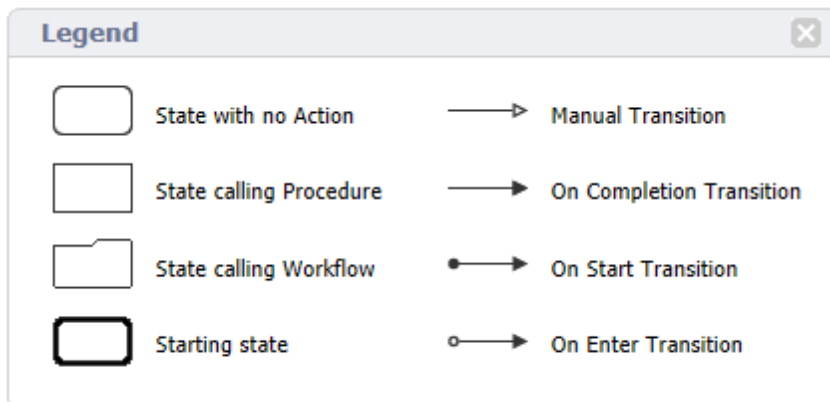
Show Legend

The following screen is another example of a workflow graph. Depending on your workflow, your graph could be very simple or much more complex than the example below.

Notice the transition line-endings and different shapes used for states.



The legend is available for reference anytime to help you become familiar with state shapes and line-ending definitions.



List view

This view will be empty until you build a workflow in the Graph view. You can view this page at any time to see a list of states and transitions in your workflow definition.

The following screen is an example of a populated Workflow Definition Details page. A **Create State Definition** link is available above the Action column, which has the same function as clicking the Create State button in the Graph view.

<div> <div>Graph</div> <div>List</div> <div>Properties</div> </div> <div>Create State Definition</div>						
Name	Type	Action	Startable	Description	Actions	
Upgrade-EndToEnd	Subjob	Upgrade-EndToEnd: Master	✓			
Success	On Completion	Target: Success				
Fail	On Completion	Target: Fail				
Success	No action					
Fail	No action					

Column Name	Description / Actions
Name	<p>This column displays state definition and transition definition names currently defined for this workflow definition.</p> <ul style="list-style-type: none"> Transition definition names are indented under a state definition name. Clicking a state definition name displays the State Definition panel (in the Graph view) for that state to see how it was defined or to make modifications. Clicking a transition definition name displays the Transition Definition panel (in the Graph view) for that transition to see how it was defined or to make modifications.
Type	<p>This is the type of action or trigger currently assigned to the state definition or transition definition.</p> <ul style="list-style-type: none"> For states - the type may show calling a subjob, subworkflow, or no action. For transitions - the trigger type is provided.
Action	<p>This is the action previously defined for the state or transition.</p> <ul style="list-style-type: none"> For states - (where the Type is "subjob") the format for actions is <code><projectName>:<procedureName></code>. Clicking the first link displays the Project Details page for the workflow project. Clicking the second link displays the Procedure Details page for the procedure used by that state. For states - (where the Type is subworkflow) the format for actions is <code><projectName>:<workflowName></code>. Clicking the first link displays the Project Details page for the workflow project. Clicking the second link displays the Workflow Definition Details page for the workflow used by that state. For transitions - clicking the Target action displays the State Definitions panel in the Graph view.
Startable	<p>A "checkmark" in this column shows this state was defined as startable, which means this state is the beginning point for your workflow.</p>

Column Name	Description / Actions
Description	A plain text or HTML description for this object.
Actions	<p>You can perform some or all of these actions:</p> <p>Create Transition - Click this link to go to the Transition Definition panel to create a new transition definition for this state.</p> <p>Access Control - Click this link to go to the Access Control page to define access privileges for this transition definition.</p> <p>Copy - Use this link to make a copy of either the state definition or the transition definition. Note: Copying a state also copies its transitions, however, you must have modify privileges on the workflow project to make a copy.</p> <p>Move - Use this link to move a state to another position in the workflow. You will be prompted to choose a state name to move this state to a location before that state. If moving a transition, note that transition can be moved only to another location for the same state—transitions cannot be moved from one state to another state.</p> <p>Delete - Use this link to delete the state definition or transition definition on the same row.</p>

Properties view

Select the Properties view button to see the Properties table. If no properties were defined for this workflow definition, no table appears.

The following screen is an example of the Properties table.

Graph	List	Properties	Create Property Create Nested Sheet Access Control	
Property Name	Value	Description	Actions	
prop1	build_1_1	beta release	Delete	
prop2	build_1_2	beta2release	Delete	

The following links and actions are available in the Properties table:

- Links and actions at the top of the table
 - **Create Property** - Click this link to go to the New Property pop-up box to create a new property for this workflow definition.

- **Create Nested Sheet** - Click this link to go the New Nested Property Sheet pop-up box to create a nested property.
- **Access Control** - Click this link to change access control privileges on the property sheet.
- Column descriptions

Column Name	Description / Actions
Property Name	Select a property name to edit that property. You can change its name, value, or add/change its description. If the property name is preceded by a folder icon, this is a property sheet. Click on the property name to open the "folder".
Value	The value currently assigned to the property.
Description	A plain text or HTML description for this object.
Actions	Delete - Click this link to delete the property on that row.

Run Workflow

Use this page to run a workflow.

- The "star" icon allows you to save this job information to your Home page.
- The "bread crumbs" Project: Upgrade-End to End / Workflow: Upgrade-Linux provide links to a previous web page.
- The name after the Run Workflow page title is the name of the workflow you intend to run.

Project: Upgrade-EndToEnd / Workflow Definition: Upgrade-Linux
Run Workflow / Upgrade-Linux ★

Starting State: Upgrade-EndToEnd
Parameters: build_name:

Starting State - This is the name of the current starting state for this workflow. If this workflow has multiple starting states and this is not the one you want to use, return to the Run Workflow pop-up dialog and select a different starting state.

Parameters -

- Any parameters previously specified for this starting start are displayed. If no parameters were defined, this area will be blank.
- If values are supplied, these are the default values specified when the parameter was created. If necessary, you can "type-over" these values to change them before running this workflow.
- You must enter a value for any blank value field labeled "Required".

Click **OK** to run the workflow when your selections are complete.

Workflow Details

This page displays workflow details, including the workflow state, project name, workflow definition name, and any properties. The following links are provided for quick access to topic sections you may want to review again.

[Graph](#)

[Show Legend](#)

[Using the graph's "quick-access" features](#)

[Show History](#)

[States](#)

[State Details panel](#)

[Parameters](#)

[Properties](#)

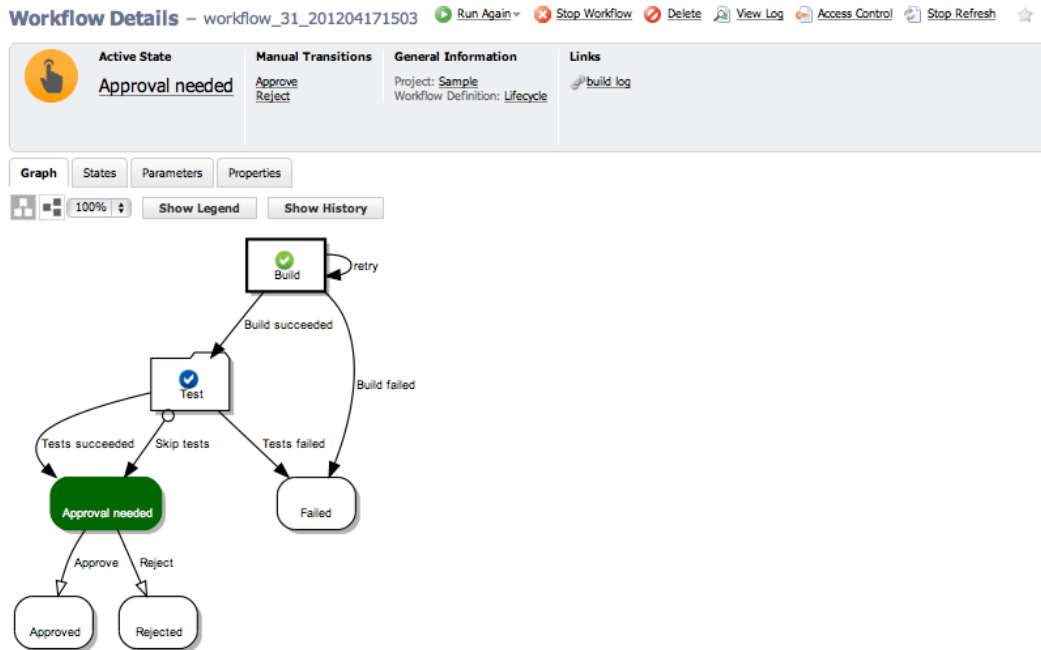
The name after the Workflow Details page title is the name of the workflow created after running a Workflow Definition. The number at the end of the workflow name is the workflow definition ID number. The ID number is auto-generated by ElectricFlow. Your workflow name will be different, depending on the information you supplied in the workflow definition.

Links and actions at the top of the table

Additional links are available if the workflow is running. When the workflow is complete, some links are no longer available.

- **Run Again** - Use this link to run the workflow definition again.
When you select the **Run Again** button, you are requesting to re-run the workflow definition with the same parameter values as the original invocation.
If you want to change the parameter values, you can select "**Run...**" from the dropdown menu (small down-arrow) next to the Run Again button.
- **Stop Workflow** - Select this option to stop the workflow. Any unfinished processes will complete, but no new transitions will occur.
- **Delete** - Deletes this workflow.
- **View Log** - Use this link to view the log created by this workflow.
- **Access Control** - Use this link to set privileges for this workflow. For more information, see the [Access Control](#) Help topic.
- **Stop Refresh** - This link is available only if a workflow is currently running. Click this link if you do not want the page to refresh the page automatically while the workflow is running.
- The "star" icon allows you to save this workflow information to your Home page.

While the workflow is running, your Workflow Details page will be similar to the following example:



Summary section - at the top of a running workflow page

- Next to the icon, notice that the state name, "Approval needed", is a link. Click this link to open the State Details panel for this state.
- The **Manual Transitions** section allows you select a manual transition if any were defined. However, if the manual transition contains any parameters whose values were not supplied at definition time, selecting that transition takes you to the Transition Workflow page to enter values.
- The **General Information** section provides links from the Project name to the corresponding Project Details page and from the Workflow Definition name to its corresponding Workflow Definition Details page.

Links

This page can include a Links section also, which will appear on the far right-side of the Summary section.

To add a link to this section:

- **Create a link to any file:**

To add a link, run a command in any job's job step using this format (this works for both local [disconnected] and non-local workspaces):

```
ectool setProperty "/myJobStep/report-urls/<myReportName>"
"/commander/jobSteps/<jobStepId>/<artifactDirectoryName>/<file-path>"
```

Example command:

```
ectool setProperty "/myJobStep/report-urls/Test Report" "/commander/jobSteps/
$/myJobStep/jobStepId]/testreport/index.html"
```

Note: If you are using this format (from ElectricCommander 4.2 and earlier):

```
ectool setProperty "/myJob/report-urls/<myReportName>" "/commander/jobs/<jobId>/<workspaceName>/<artifactDirectoryName>]/<file-path>"
```

Example command:

```
ectool setProperty "/myJob/report-urls/Test Report" "/commander/jobs/${myJob/jobId}/  
${myJobStep/workspaceName}/testreport/index.html"
```

It will continue to work only for non-local workspaces. We recommend, however, that you change the command to use the newer format.

- **Create a link to any directory:**

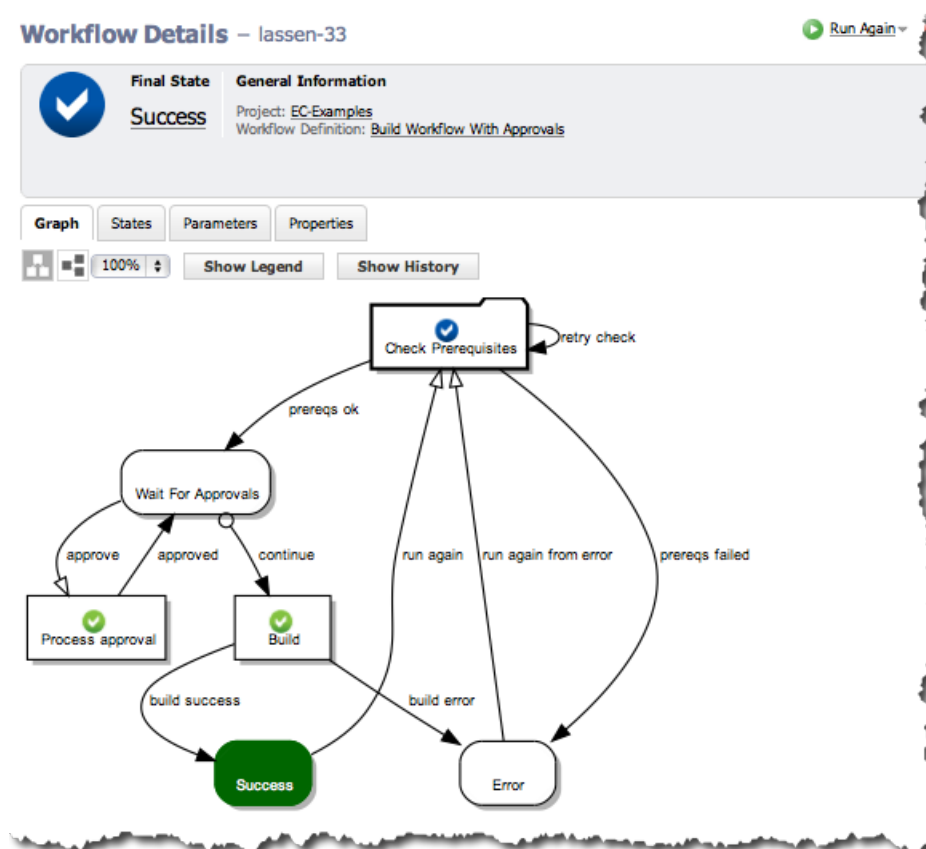
To add a link, run the following style command in any job's job step:

```
ectool setProperty "/myJobStep/report-urls/Main Job Workspace"  
"file:///WinStor2/scratch/chron55build/${myJob/jobName}"
```

Note: Links to a directory automatically work in Internet Explorer, but if using Firefox, local links are disabled unless the default security policy is modified or an extension is used. Refer to http://kb.mozillazine.org/Links_to_local_pages_don%27t_work for more details.

When the workflow is complete...

Your Workflow Details page will be similar to the following example:



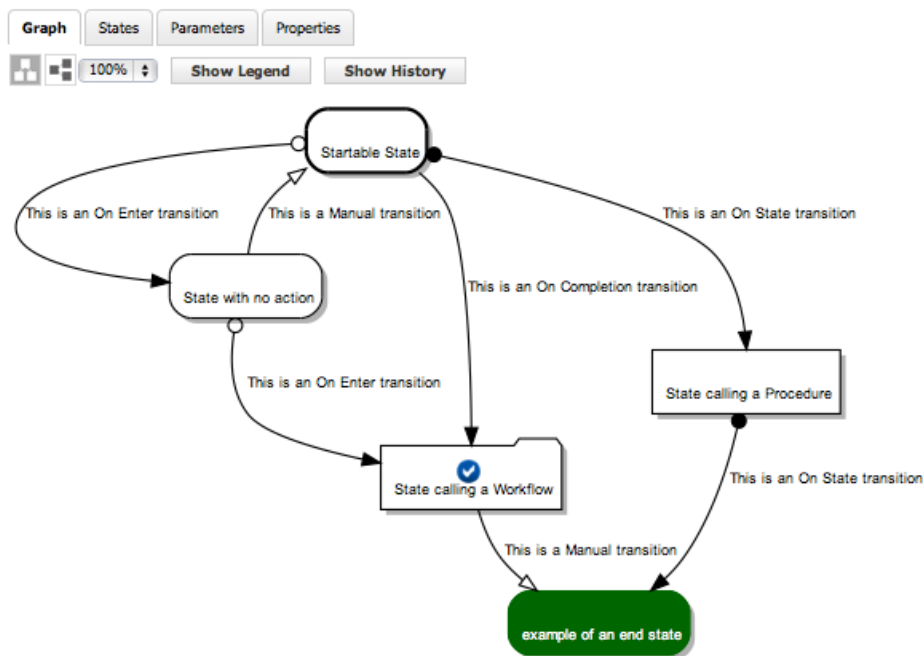
- When the workflow is complete, fewer links are available at the top of the table and the Manual Transitions section is no longer available.
- The "checkmark" icon indicates the workflow is complete. Click the state shown under Final State ("Success" in our example), to open that state's State Details panel.
- The **General Information** section is the same as when the workflow was running.

Graph tab

The Graph tab displays your workflow in a visual format. This format is the default view. The following screen is an example of a workflow graph. Depending on your workflow, your graph could be very simple or much more complex than the example below.

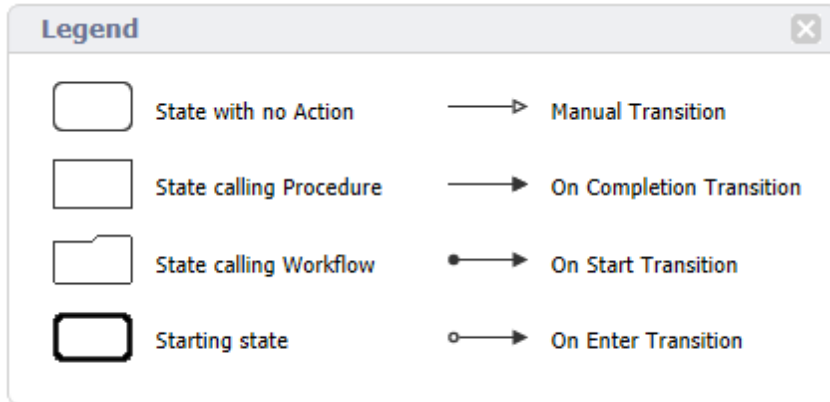
Note: This graph updates in real-time, which means you will see the "active" state highlighted with a color. An icon on the state represents an action.

Use the icons below the Graph tab to rotate the graph for better use of your screen viewing area, and you can re-size the graph using the "%" drop-down menu to choose another size.



Show Legend tab

The graph above illustrates all state and transition types. The state and transition names in this graph are labels for the type of state or transition represented. Use the Show Legend tab to see a quick reminder of state shape and transition line-ending definitions.



Using the graph's "quick-access" features

The following screen example is a portion of a workflow graph.

In the graph, states and transitions are links

- Clicking on a state takes you to the State Details panel to that state's transitions, parameters, and properties.
- Clicking on a transition takes you to the Transition Details panel to see details specifically about that transition, including its condition information.
(Hover your mouse over the transition or transition name and click when the color changes.)

However, if you **right-click** on a state or transition, other link options are available. These options are different for a running or completed workflow versus the options on the Workflow Definition Details page that were available to you while you were defining your workflow.

Available pop-up context-sensitive links include:

- For states, depending on which actions were defined for the state, you may have one or more of the following options:
 - Go to Action - opens the Workflow Details or Job Details page for that state.
 - Go to Action Definition - for states calling a procedure, the Procedure Details page is opened. For states calling a workflow, the Workflow Definitions Details page for the "subworkflow" is opened.
 - Go to State Definition - opens the State Definition panel for that state.
- For transitions, the only link available is: Go to Transition Definition, which opens the Transition Definition panel.

Show History

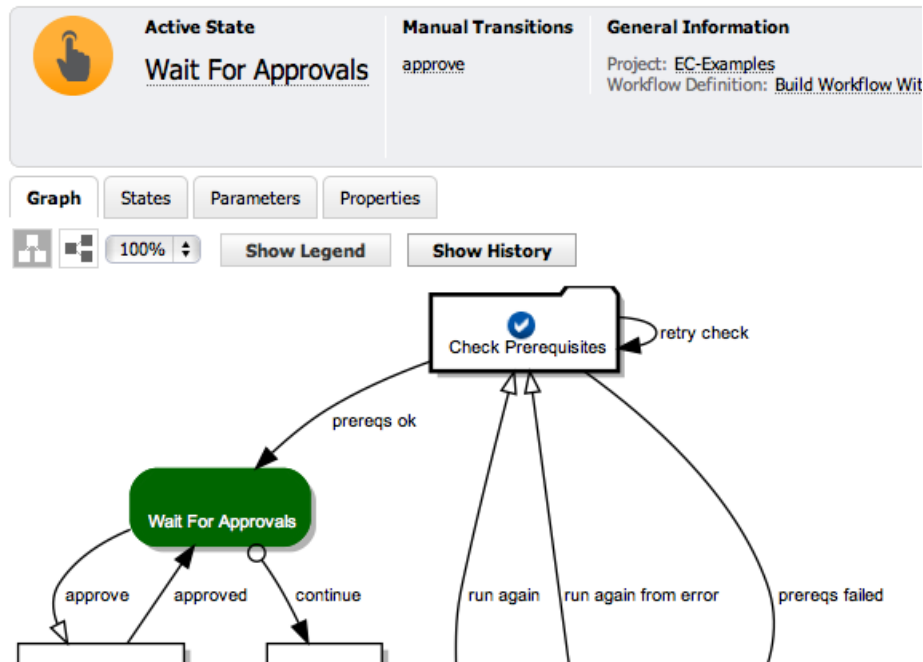
Selecting Show History allows you to see at-a-glance how your workflow progressed through its states and transitions, or for example, how many times the same state was visited and depending on outcome, which transitions were taken and then the outcome of those target states.

The following example shows a workflow started with the "Check Prerequisites" state and is now at the "Wait For Approvals" state.

Project: EC-Examples

Workflow Details – lassen-139

[Run Again](#) [Stop Workflow](#)



So how did the workflow progress through these states?

In the next screen example, with the History panel open, you can clearly see the workflow started and the check failed the first time, so the "retry check" transition was taken. The subworkflow was successful the second time, and the workflow has now transitioned to the "Wait For Approvals" state.

Project: EC-Examples

Workflow Details – lassen-139

Run Again Stop Workflow Delete View Log Access Control Stop Refresh

Active State
Wait For Approvals

Manual Transitions
 approve

General Information
 Project: EC-Examples
 Workflow Definition: Build Workflow With Approvals

Graph States Parameters Properties

100% Show Legend Hide History

History

- Check Prerequisites workflow-check-140 : Error
- retry check
- Check Prerequisites workflow-check-141 : Success
- prereqs ok
- Wait For Approvals

Notice the links displayed in the History panel. You can select links for completed jobs or workflows, which display the Job Details or Workflow Details page, respectively.

States tab

Selecting the States tab provides a table containing all state names and actions for this workflow.

Workflow Details – workflow_2044_201204291458

Run Again Delete View Log Access Control

Final State
Approval needed

General Information
 Project: Sample
 Workflow Definition: Lifecycle

Graph **States** Parameters Properties

State Name	Action
Build	BuildProduct-build-85304 [Show All]
Test	workflow_2045_201204291458 : System tests [Show All]
Failed	
Approval needed	

The following links are available in the States table

- State Name - Click any state name in this column to go to that state's State Details panel. (See the next section for more information about the State Details panel).
- Action - Actions are available in this column only if you are using a subjob or a subworkflow. If so, links in this column connect you to the Job Details or Workflow Details page for the subjob or subworkflow, respectively.
- [Show All] link - If the state executed multiple times during a workflow run, only the last subjob or subworkflow is displayed. However, clicking on Show All displays the results for all subjobs or subworkflows executed during a workflow run.

State Details panel

This panel displays state components, including transitions, parameters, and properties.

- Use the **Access Control** link to set privileges for this state. *For more information*, see the [Access Control](#) Help topic.
- The state name, "Build" (in our example), is shown at the top of the panel.
- Job or Workflow status links to the job or workflow created by the action.
- The Transitions tab is open to display the transition table listing all transitions for this state.

Column descriptions

Column Name	Description / Actions
Name	One or more names of transitions currently defined for this state. Click on a transition name to open the Transition Details panel.
Trigger	The type of trigger currently defined for this transition.
Target	The target state for this transition. Click on a target state name to open the State Details panel for that state.
Condition	This condition was specified when the transition was created. A condition specifies under which circumstances the transition is taken. You can edit this condition on the Transition Definition pane, but if you edit the condition during a running workflow, the change will not take effect until you run the workflow again.
Actions	Access Control - Use this link to set permissions on this transition.

State Details

Build

[Access Control]

No Description

BuildProduct-build-85304

[Show All]

Transitions

Parameters

Properties

Name	Trigger	Target	Condition	Actions
Build failed	onCompletion	Failed		Access Control
Build succeeded	onCompletion	Test	<code>\$/javascript mySubjob.outcome == "success";]</code>	Access Control
retry	onCompletion	Build	<code>\$/javascript mySubjob.outcome == "error" && myState.getProperty("/increment numberOfRetries") < 3;]</code>	Access Control

The **Parameters** tab provides a table listing all defined parameters for this state. You will see any parameters passed to this state the last time this state was entered, but only if parameters were created on the State Definitions Details panel. This table does not allow you to edit the parameter's value.

These parameters are copied automatically to the top-level property sheet and you will see them in the properties table also.

State Details

Build

No Description

BuildProduct-build-85304 [Show All]

Transitions

Parameters

Properties

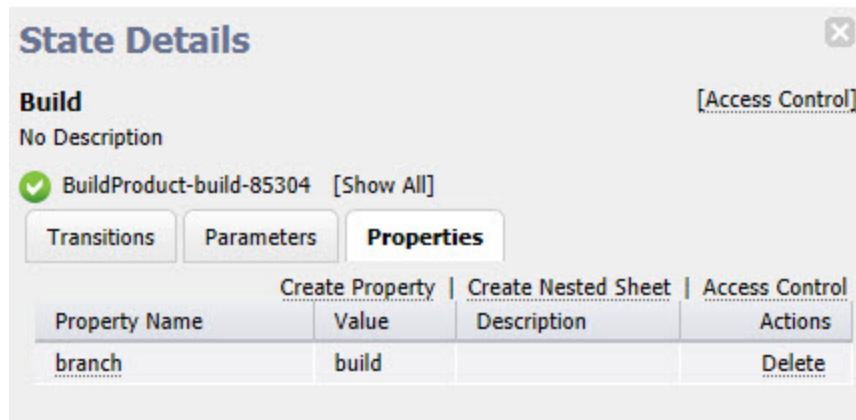
Name	Value
branch	build

Column descriptions

Name - The name of the parameter defined for this state.

Value - The value currently defined for this parameter.

The **Properties** tab provides a table containing all defined properties for this state. If no properties were defined for this state definition, this table will be blank. All defined parameters are automatically copied to the top-level property sheet. Any custom properties defined for the state will be included in this table also.



Links at the top of the table

- **Create Property** - Click this link to go to the New Property pop-up box to create a new property for this state.
- **Create Nested Sheet** - Click this link to go the New Nested Property Sheet pop-up box to create a nested property sheet.
- **Access Control** - Click this link to add or decrease access control on the property sheet.

Column descriptions

Column Name	Description / Actions
Property Name	Select a property name to edit that property. You can change its name, value, or add/change its description. If the property name is preceded by a folder icon, this is a property sheet. Click on the property name to open the "folder".
Value	The value currently assigned to the property.
Description	A plain text or HTML description for this object.
Actions	Delete - Click this link to delete the property on that row.

Parameters tab

This tab provides a table listing all defined parameters for this workflow's starting state. You will see any parameters passed to the starting state when this workflow was launched, but only if parameters were created on the State Definitions Details page. This table does not allow you to edit the parameter's value.

These parameters are copied automatically to the top-level property sheet and you will see them in the properties table also.

Name	Value
test1	1
test2	2

Column descriptions

Name - The name of the parameter defined for this starting state.

Value - The value currently defined for this parameter.

Properties tab

Select the Properties tab to see the Properties table. If no properties were defined for this workflow definition, this table will be blank.

Create Property Create Nested Sheet Access Control			
Property Name	Value	Description	Actions
property1	full build		Delete
property2	test only		Delete
test1	1		Delete
test2	2		Delete

The links at the top of the table

- **Create Property** - Click this link to go to the New Property pop-up box to create a new property for this state.
- **Create Nested Sheet** - Click this link to go the New Nested Property Sheet pop-up box to create a nested property sheet.
- **Access Control** - Click this link to add or decrease access control on the property sheet.

Column descriptions

Column Name	Description / Actions
Property Name	Select a property name to edit that property. You can change its name, value, or add/change its description. If the property name is preceded by a folder icon, this is a property sheet. Click on the property name to open the "folder".
Value	The value currently assigned to the property.
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code> .
Actions	Delete - Click this link to delete the property on that row.

Workflow Log

This page displays the workflow log.

Notice the name after the page title, "template-1257" — this is the name of the workflow, which includes the workflow name plus the ElectricFlow auto-generated ID number, and it is the object of the workflow log in our example.

Links and actions above the table

- Drop-down menu - Use the down-arrow to select Error, Warn, or Info. The first time you choose a severity level, it will become your default level—the one you always see first when you view this page. Selecting another level changes your default view.
- The "star" icon allows you to save this workflow definition to your Home page.
- The "bread crumbs" *Project: SoftwareBuild / Workflow: template-1257* provide links to a previous web page.

Project: SoftwareBuild / Workflow: template-1257

Workflow Log – template-1257

INFO

7 Results

Time	Severity	User	Subject	Message
2010-12-06 15:12:42 PST	INFO	admin	wip	User 'admin' has completed the workflow
2010-12-06 15:12:28 PST	ERROR	project: SoftwareBuild	wip	Failed to start the sub-job for state 'wip': Missing parameter(s) to 'wf_start_wip': plugin, version
2010-12-06 15:12:28 PST	INFO	project: SoftwareBuild	wip	Sending onEnter email notifiers for state 'wip'
2010-12-06 15:12:28 PST	INFO	admin	start_work	User 'admin' is taking manual transition 'start_work' from state 'stable' to state 'wip'.
2010-12-06 14:55:30 PST	INFO	project: SoftwareBuild	stable	Sending onEnter email notifiers for state 'stable'
2010-12-06 14:55:30 PST	INFO	project: SoftwareBuild	stable	Starting state: 'stable'
2010-12-06 14:55:30 PST	INFO	admin	workflow_53	Created workflow 'workflow_53'

Records per page: 20 1 thru 7 of 7

Column descriptions

Time - Displays the time when an event occurred that caused the server to generate a message.

Severity - the three severity levels are:

- **ERROR** - An unexpected failure was encountered while entering a state or launching a sub-action. Generally, this error indicates a critical problem with the workflow that requires fixing the workflow definition.
- **WARN** - A non-critical issue was encountered while the workflow was running.
- **INFO** - Provides workflow activity information including the state entered, transitions taken, and so on.

User - The name of the user or project principal that explicitly launched the job. This property is blank when the job is launched by a schedule.

Subject - Objects displayed in this column are the subject of the message. These objects are linked to either the Workflow Details or the State Details page.

Message - A text message generated by the ElectricFlow server while the workflow was running.

Transition Workflow

Use this page to transition your workflow to a different startable state.

Note the name after the page title, "template-99"—this is the name of the workflow in our example.

Links and actions above the table

- The "star" icon allows you to save the Transition Workflow page to your Home page.
- The "bread crumbs" *Project: SoftwareBuild / Workflow: template-99* provide links to a previous web pages.

Project: SoftwareBuild / Workflow: template-99

Transition Workflow – template-99

Transition: start_work
State: stable
Target State: wip
Parameters version: Required

OK Cancel

Field descriptions

Enter information in the fields as follows:

Field Name	Description
Transition	The name of this transition.
State	The name of the state that owns the transition— the active state of the workflow.
Target State	The name of the target state— the state where the workflow will transition.
Parameters	In the screen example above, "version" is the name of the parameter and you must enter a value for this parameter. This is a required field.

Click **OK** to transition the workflow to the target state.

Workspaces

This page displays all workspaces currently available to ElectricFlow. At a glance, you can see all paths to every workspace.

- To create a new workspace, click the **Create Workspace** link.
- To find a workspace, use the **New Search** link.

Column descriptions

Column Name	Description / Actions
Workspace Name	The name created for this workspace. Click on a Workspace Name in this column to edit that workspace.

Column Name	Description / Actions
Zone	The name of the zone where this workspace resides.
Enabled	This workspace is enabled if this box is "checked". Note: When this box is checked, the workspace is enabled, which means it can be accessed. In the case where a job or job step cannot access the workspace, the job will "queue", waiting for the workspace to become available.
Local	If the box is "checked", the workspace is "local", which means files in the local workspace are accessible only from the resource that originally created the file.
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code> .
Drive Path / UNC Path / UNIX Path	The path to the workspace.
Actions	Copy - Use this link to make an exact duplicate of an existing workspace, then select the copy to go to the Edit Workspace page to change the workspace name. Delete - Use this link to delete the workspace on the same row.

Note: Add access control privileges to a workspace on the Edit Workspace page.

Workspace - create new or modify existing workspace

To create a new workspace

Enter information in the fields as follows:

Field Name	Description
Name	Enter a unique name to distinguish this workspace from other workspace names.

Field Name	Description
Description	A plain text or HTML description for this object. If using HTML, you must surround your text with <code><html> ... </html></code> tags. The only HTML tags allowed in the text are: <code><a></code> <code></code> <code>
</code> <code><div></code> <code><dl></code> <code></code> <code><i></code> <code></code> <code></code> <code><p></code> <code><pre></code> <code></code> <code><style></code> <code><table></code> <code><tc></code> <code><td></code> <code><th></code> <code><tr></code> <code></code>
Enter three paths to use on resources to access the workspace root directory:	
UNC Path	A path in UNC notation (such as <code>//server/ec/a/b/c</code>) to use on Windows machines. The path can use either slashes or backslashes for separators.
Drive Path	A second path for Windows machines, based on drive notation such as <code>N:</code> or <code>N:/b/c</code> . If the drive path contains more than just a drive letter, additional directory names must match the trailing directories of the UNC path. For example, if the UNC path is <code>//server/ec/a/b/c</code> , then <code>N:/b/c</code> can be used as the drive path (agents automatically map <code>N:</code> to <code>//server/ec/a</code>), but <code>N:/x/c</code> is not allowed.
UNIX Path	The path to use for UNIX machines, typically based on an NFS mount. You must ensure appropriate mounts were created on all machines where the path will be used. <i>ElectricFlow does NOT automatically mount filesystems for UNIX.</i>
<p>Note: If the workspace will be used only on UNIX machines, you can omit the UNC and drive paths. Conversely, if the workspace will be used only on Windows machines, you can omit the UNIX path.</p> <p>For more information, see the Workspaces and Disk Space Management Help topic.</p>	
Zone	Use the drop-down arrow to select a zone for this workspace.
Local	Select this checkbox if the workspace is "local", which means files in the local workspace are accessible only from the resource that originally created the file.
Enabled	<p>"Check" this box to enable this workspace.</p> <p>When this box is checked, the workspace is enabled, which means it can be accessed. In the case where a job or job step cannot access the workspace, the job will "queue", waiting for the workspace to become available.</p>
Credential Project and Credential Name	If you need to enter a user name and password to mount the workspace, you must enter the credential project name and then choose the appropriate credential name, which contains the required user name and password. Click the Browse link to select information for these fields.

Click **OK** to save your workspace specification and see it listed on the Workspaces page.

If you need to add properties or privileges to this workspace:

Click **OK** to create the workspace and return to the Workspaces page, then click on the workspace name to go to the Edit Workspace page.

To edit an existing workspace

All previously supplied workspace information is available on this page and you can modify existing information or add new properties or privileges.

- Click the **Access Control** link [at the top of the page] to add access privileges to this workspace.
- To assign a new workspace name, highlight the existing name and type-in a new name.
- Re-assign paths if necessary.
- Use the drop-down menu to select a different zone if necessary.
- Add or modify the Credential Project name or the Credential Name if necessary.

Click **OK** to save your modified information.

If Custom Workspace Properties were already assigned, you may modify these properties or add new ones. If a property name is preceded by a "folder" icon, this denotes a Nested Sheet.

Click the "star" icon at the top of the page to add this page to your Shortcut section on the Home page.

For more workspace information, see the [Workspace and Disk Space Management](#) Help topic.

Workspace File

This page displays a file from the workspace for a job, typically the log file for a job step. For log files, this file contains both standard output and standard error from the step.

If you view a file while the job step generating the file is still running, the page refreshes automatically as information is added to the file.

Tip

If the log does not contain the output that you were expecting, check the job step commands. If the step redirects its output to a different location, this information will not appear in the log file.

ElectricFlow Platform Objects and Functionality

All the topics in this section are about a particular object or functionality in the ElectricFlow platform. They are the same as the topics in the Help system on the automation platform UI.

From the automation platform UI, you can access the topics as follows:

- Direct access to these Help topics is from the Help system Table of Contents. Although these topics are not directly linked within the ElectricFlow product, many "page-specific Help topics contain links to one or more of these topics to provide more detailed information quickly.

- To access these Help topics while using ElectricFlow, click the **Help** link at the top-right corner on any automation platform web page.

Access Control

[Privileges](#)

[Users and Groups](#)

[Special users and groups](#)

[Access Control Lists - allow and deny](#)

[Inheritance](#)

[System Objects](#)

[Access Control and Jobs](#)

[Examples for Increased Security](#)

Overview

ElectricFlow provides a comprehensive mechanism to control how individuals use the system.

- Users must log in to view information or perform operations.
- After you log in, system access is limited based on:
 - user name
 - the groups to which that user belongs
 - permissions specified for various ElectricFlow objects

Access Control is the ElectricFlow functionality that provides security for all system objects. **After** you are familiar with the following information describing the Access Control system, review the two examples, Basic ACL Setup and Team ACL Setup (at the end of this Help topic), which may provide guidance or insight for how you might setup enhanced ElectricFlow security at your site.

Privileges

ElectricFlow supports four privilege types for each object:

- **Read** - Allows object contents to be viewed.
- **Modify** - Allows object contents (but not its permissions) to be changed.
- **Execute** - If an object is a procedure or it contains procedures (for example, a project), this privilege allows object procedures to be invoked as part of a job. For resource objects, this privilege determines who can use this resource in job steps.
- **Change Permissions** - Allows object permissions to be modified.

Users and Groups

ElectricFlow uses account information from multiple sources. In most cases, the primary account information source is an external LDAP or Active Directory repository:

- Both user and group information is retrieved from the repository
- Local users and groups can be defined within ElectricFlow

Note: To view user and group information, and to modify local user and group information, click the Administration tab and select either the Users or Groups subtab. External account information **cannot** be modified from ElectricFlow.

If the same user exists in multiple sources, only the highest priority name is used. A priority order is defined among external repositories, but local names have the highest priority. Thus, if you define a local user with the same name as an LDAP user, you will effectively mask the user in the LDAP account.

For local user accounts, only local groups are considered—group information from external repositories is not used. For accounts from a particular repository, groups from that repository are used along with local groups, but groups in other repositories are not considered.

In other words, you can define local groups in ElectricFlow to supplement groups defined in external repositories. When you view information about users in ElectricFlow, only relevant groups are shown. For example, when you view group information for a local user, only local groups are displayed.

Note: Groups are managed by name only, without regard to source. If a particular group name exists in different repositories, there is no way to distinguish between these groups inside ElectricFlow. Access given to one group is the same for any other group with the same name.

Special Users and Groups

The **admin** local user has special significance. If you are logged in as "admin," you automatically have all privileges on all objects, regardless of any other system settings. This privilege set is a fallback mechanism in the event too many privileges get removed for an object, leaving it unusable.

The admin account cannot be deleted.

If the admin account is missing, ElectricFlow will recreate the account the next time it starts up with password "changeme."

The **Everyone** group is predefined by ElectricFlow and cannot be deleted. Every user is automatically a member of the Everyone group.

For each project, ElectricFlow uses the project name and automatically defines a user associated with that project, called the project principal. For example, the **project principal** for a project named "nightly builds" is "project: nightly builds" (notice that there are two spaces in this name)—this principal is used for jobs running under the project, as described in the ["Access control and jobs"](#) section later in this Help topic.

Access Control Lists - allow and deny

Each ElectricFlow object, such as a project, procedure, job, property sheet, workspace, schedule, or resource contains an access control list (ACL). Individual properties do not have their own access control, but instead use

access control lists from their parent containers.

To view the access control list for an object, go to the web page displaying the object and click the **Access Control** link. An access control list can contain any number of entries— each entry names a particular user or group and indicates *allow*, *deny*, or is blank for each of the four privileges.

To determine whether a user can perform an operation on a particular object, ElectricFlow determines which of the four privileges is required for that operation, then searches all access control entries that refer to that user or groups containing that user. To be allowed access, at least one of the matching entries must specify "allow" and none of the entries can be "deny." A "deny" entry overrides an "allow" entry in the same ACL.

Inheritance

If an object's access control list does not indicate what to do for the user who is trying to access it, ElectricFlow looks in the access control list for the object containing the target object, then its parent, and so on up to a top-level object covering the entire server. This mechanism is called *inheritance*.

For example, a nested property sheet on a procedure inherits access control information from its parent property sheet, the procedure, the project containing the procedure, and the server. When you view the object access control list, you also see inherited access control lists so you can trace the object's inheritance chain.

When ElectricFlow performs an access check on an object, it begins with the access control list for that object.

- If a "deny" entry matches an entry in that list, access is denied.
- If there is no "deny" entry, but a matching "allow" entry is found, access is allowed.
- If there is no matching entry that specifies either "allow" or "deny," ElectricFlow moves to the next access control list in the inheritance chain and repeats the process.
- A matching entry in a lower-level access control list takes priority over entries in higher-level access control lists. If no matching entry in any list is found, access is denied.

Inheritance allows you to control access to a collection of objects because you can make changes in just one place. When new objects are created, their access control lists are empty, so all access control for the entire system is determined by the server-level list.

- Typically, when a new project is created, you define an access control list for the project and allow everything in the project to inherit from the project.
- If you do not define additional access control information in project components, all project access is determined by the project's access control list.

In some situations you may want tight control over object access, so changes in parent objects do not affect access to the object or its children. To achieve this, you can "break inheritance," which means a particular object does not inherit from its ancestors. The ElectricFlow web interface allows you to break and restore inheritance for any object on which you can change permissions. However, ***if you break an object's inheritance, it affects all children of that object.***

Additional Information about "deny"

If a user matches a deny rule, however indirectly, the user is denied access. Even if a specific ACL entry granting (for example) read access by user name exists, a less specific deny ACL entry overrides that specific

rule. Therefore, avoid "deny" rules if at all possible (especially to groups), or you may experience some unexpected permissions issues.

You can, however, enforce an implicit "deny" by allowing an object to fall off the end of the inheritance chain. This practice allows denial behavior without the risks associated with an explicit ACL entry to deny access.

IMPORTANT: If you break inheritance on an object with an empty access control list, the object will become completely inaccessible. You will not be able to restore inheritance because you no longer have the right to change permissions on that object. If this happens, you must contact your system administrator who can login as admin and restore inheritance.

Consider ElectricFlow principals (actors) from two perspectives:

- A running job or workflow runs using the project as the identity, so if a job requires special permissions you must use the project as the principal in the ACL entry.
- Users and groups come into play when humans manipulate the GUI or execute ectool commands from the command line (or via a shell script) -- basically anywhere that a script or human needs to log in to ElectricFlow. Having logged in, a group and id exist, which can be used in ACL entries.

For example, consider a property that will store a build number. You can set an ACL that breaks inheritance (implicit deny), and then add an ACL entry granting permissions for the project to be able to increment the property value. This prevents humans from manipulating the build number but permits a running job to increment the build number at will.

Notes:

- When it is practical, a job will run with multiple identities—the project and the user that launched the job. You cannot, however rely on this behavior for security because jobs run from a schedule (such as CI jobs) and jobs started by a workflow do not have a user principal, just the project principals.
- A job can have multiple project principals because it "accumulates" new project principals when it calls into another project's subprocedure (and then releases the other project's identity when the subprocedure returns). This makes it possible to create trusted libraries.

System Objects

A few special *system objects* contain access control lists related to overall ElectricFlow system administration. These access control lists are available from the Administration page. The system objects are:

- **Server** - an ElectricFlow system top-level object. Every other object in the system is contained in the server object and inherits access control information from the server object unless inheritance is broken.
- **Administration** - *Read* permission allows access to the `getStatus`, `getDatabaseConfiguration[s]`, `getEmailConfig[s]`, and `export(global)` API functions.
Modify permission allows access to the `shutdown`, `setDatabaseConfiguration`, `createEmailConfig`, `deleteEmailConfig`, `modifyEmailConfig`, and `import(global)` API functions.
- **Artifacts** - *Read* permission allows access to the `getArtifact` API functions.
Modify permissions allows access to `createArtifact` and `deleteArtifact` functions.

- **Directory - Read** permission allows access to the `getUsers`, `getGroups`, and `getDirectoryProviders` API functions.
Modify permission allows access to the `createUser`, `createGroup`, `deleteUser`, `deleteGroup`, `createDirectoryProvider`, `modifyDirectoryProvider`, `deleteDirectoryProvider`, `testDirectoryProvider`, and `moveDirectoryProvider` API functions.
- **Email Configurations - Modify** permission allows access to the `createEmailConfig` and `deleteEmailConfig` API functions.
- **Force Abort - Execute** permission controls access to the `--force` flag on `abortJob`. By default, the ACL is created with Everyone: execute permission in addition to inheriting from the "Server". To force abort a job, the user must have execute permission on the job and execute permission on the `forceAbort` ACL.
- **Logging - Modify** permission allows access to the `logMessage` API function.
- **Licensing - Read** permission allows access to the `getLicense[s]` API functions.
Modify permission allows access to the `importLicenseData` and `deleteLicense` API functions.
Execute permission allows access to the `getAdminLicense` API function.
- **Plugins - Modify** permission allows access to the `createPlugin`, `deletePlugin`, `installPlugin`, `uninstallPlugin`, `promotePlugin` API functions, and the `modifyPlugin` API function requires modify permission on the target plugin. For `getPlugin`, *Read* permission is required on the target plugin.
- **Priority - Execute** permission allows the user who launches a procedure (using the `runProcedure` API function) to raise the priority of the job.
- **Projects - Modify** permission allows access to the `createProject` and `deleteProject` API functions.
- **Repositories - Read** permission allows access to the `getRepository` API function.
Modify permission allows access to the `createRepository`, `deleteRepository`, `modifyRepository`, and `moveRepository` API functions.
- **Resources - Modify** permission allows access to the `createResource` and `deleteResource` API functions.
- **Session - Execute** permission allows access to the `login` API function.
- **Workspaces - Modify** permission allows access to the `createWorkspace` and `deleteWorkspace` API functions.
- **ZoneAndGateways - Modify** permission allows access to the `createZone` and `deleteZone` API functions. *Modify* permission also allows access to the `deleteResource` API function when the resource belongs to a Gateway.

Access Control and Jobs

When a job executes, it usually needs to access objects in ElectricFlow. For example, a job step command may refer to a parameter value, which is a property associated with the job object, or a step may invoke *ectool* to modify properties or any other ElectricFlow state. This process leads to questions:

- ***Under which user name does the job execute?***

Procedures always run under the project principal user ID for the project that contains the procedure. If a procedure invokes a subprocedure in another project, that subprocedure will run under its own project's project principal and the project principal of its calling procedure. When a procedure is running under

multiple project principals, its steps will perform any operations for which any one of its project principals allow.

- ***How does ElectricFlow initialize job permissions when the job starts?***

This question pertains to job object permissions. When a job starts, ElectricFlow sets full access control entries on the job for the project principal and the user who launched the job— assuming the job was launched by a user and not a schedule.

- ***What permissions are needed to abort a job?***

Aborting a job requires *execute* permission on the job. If a job is launched by a user, that user is given all privileges on the job. If a job is launched by a schedule, the schedule's execute privileges are copied to the job.

The access control system determines whether jobs can be executed or not.

- For a user to run a job without creating a schedule, the user must have *execute* permission on the top-level procedure being executed.
- To create a schedule to run a procedure, a user must have *modify* permission for the project containing the schedule. After a schedule is created, no additional permissions are required to start jobs under the auspices of that schedule.

Examples for Increased Security

Initially, the *Everyone* group had Read and Execute permission on all ElectricFlow objects by default. The following examples illustrate how you might customize ElectricFlow default Access Control settings for your organization.

Caution: Make sure you are familiar with the Access Control system before making changes. Incorrectly setting ACLs can severely impact ElectricFlow behavior.

Abbreviations used in the examples

A = Allow

I = Inherit (some places referred to as "Don't Care")

D = Deny

Change Permission = Change

Example 1 - Basic ACL Setup

This example illustrates how you might change Access Control defaults to set up a basic level of security. The easiest way to manage ACLs is by using different ElectricFlow groups that allow users to perform specific roles.

For our example, we will create and use the groups called EC-administrator and EC-designer, and we will modify the default *Everyone* group. (When setting up your groups, you would choose any group names that suit your organization, but note that the "EC-" prefix is reserved for Electric Cloud use only.) This is how we want to define our groups:

- **EC-administrator group**
This group has most of the power and abilities of the 'admin' user login, but allows for more flexibility.
- **EC-designer group**
This role is more significant. Users in this group can create and modify most ElectricFlow objects.
- **Everyone group**
This predefined group is used for people who only need to run ElectricFlow procedures, and have no need to create or modify them.

As you begin to modify the Access Control system, note that the system has some project principal ACEs that should remain in place for correct system operation.

Server ACLs

The Server is the foundation for almost all ACL checks. Generally, every other object in the system inherits these privileges. For this basic ACL configuration we want to set Server ACLs this way:

Type	Name	Read	Modify	Execute	Change
group	EC-administrator	A	A	A	A
group	Everyone	I	I	I	I

Custom Server Properties ACLs

An often overlooked ACL setting is Server Property Sheet. Allow the Everyone group Read permission on this object.

Type	Name	Read	Modify	Execute	Change
group	Everyone	A	I	I	I

System Object ACLs

There are 14 categories of System objects with ACLs.

1 - Administration – allow the EC-designer group Read privileges.

Type	Name	Read	Modify	Execute	Change
group	EC-designers	A	I	I	I

2 - Artifacts – allow the EC-designer group Read privileges.

Type	Name	Read	Modify	Execute	Change
group	EC-designers	A	A	I	I
group	Everyone	A	I	I	I

3 - Directory – allow the Everyone group Read privileges.

Type	Name	Read	Modify	Execute	Change
group	EC-designers	A	I	I	I

4 - Email Configurations – allow the Everyone group Read and Execute privileges.

Type	Name	Read	Modify	Execute	Change
group	Everyone	A	I	A	I

5 - Force Abort - no changes needed for this category.

6 - Logging - no changes needed for this category.

7 - Licensing - no changes needed for this category.

8 - Plugins - allow the Everyone group Read privilege.

Type	Name	Read	Modify	Execute	Change
group	Everyone	A	I	I	I

9 - Priority - no changes needed for this category.

10 - Projects - allow the Everyone group Read and Execute privileges, and the EC-designer group everything except Change privilege.

Type	Name	Read	Modify	Execute	Change
group	EC-designers	A	A	A	I
group	Everyone	A	I	A	I

11 - Repositories - allow the Everyone group Read and Execute privileges.

Type	Name	Read	Modify	Execute	Change
group	Everyone	A	I	I	I

12 - Resources - allow the Everyone group Read and Execute privileges.

Type	Name	Read	Modify	Execute	Change
group	Everyone	A	I	A	I

13 - Session - allow the Everyone group Execute privileges.

Type	Name	Read	Modify	Execute	Change
group	Everyone	I	I	A	I

14 - Workspaces - allow the Everyone group to have Read and Execute privileges.

Type	Name	Read	Modify	Execute	Change
group	Everyone	A	I	A	I

Note: Sites that want a slightly more open system might consider granting the "EC-designers" group Modify access to Resources and Workspaces.

Plugin Project ACLs

Generally, you will want to allow the Everyone group Read privileges on every plugin you expect to use. (Each plugin has its own name, which is the project name also.)

Type	Name	Read	Modify	Execute	Change
group	Everyone	A	I	I	I

Project ACLs

Next are special Electric Cloud projects, Default, EC-Utilities and EC-Examples.

Projects: Default, EC-Utilities, EC-Examples

Use caution when setting privileges on the EC-Utilities project. These are powerful procedures and some have their own ACL settings. Remember the 'admin' user will always be able to use these utilities. Allowing the EC-administrator group full privileges accomplishes the same goal for users in that group. We allow the Everyone group Read only privileges. This is a unique project because it does not inherit ACLs from anywhere else (for example, Server or Projects).

Type	Name	Read	Modify	Execute	Change
group	EC-administrator	A	A	A	A
group	Everyone	A	I	I	I

Example 2 - Team ACL Setup

Some organizations may want to have two or more different and separate teams use ElectricFlow at the same time. In this example, people on one team have little knowledge about the other team who uses ElectricFlow and no direct access to any objects used by the other team. Our "Team" example describes how to modify default ElectricFlow Access Control settings to create a multi team security configuration using ACLs.

To divide your ElectricFlow system for use by separate teams, you will want to create one or more separate Projects for each team. You may decide to share some projects across teams.

Create your teams (groups). For our example, we will begin with two teams, T1 and T2. Each team has a designer group and a user group. Using groups allows the flexibility to have a designer from one group work as a user in another group.

- Create the EC-administrator group
This group has most of the power and abilities of the 'admin' user login, but allows for more flexibility. Assigning people to the EC-administrator group allows better tracking of ElectricFlow administrative activity.
- Create two designer groups: T1-designer and T2-designer
Members of the designer groups will have the ability to create and modify procedures and other ElectricFlow objects for their team. A member of the designer group from one team will not be able to view or use the objects of the other team.
- Create two user groups: T1-user and T2-user
Members of the user groups will have the ability to run procedures that belong to their team. A member of the user group from one team will not be able to view or use objects that belong to the other team. who are not in any other group. This setup allows other users to access ElectricFlow without exposing the work of any of the teams. A person not in a group for either team will not be able to see objects of either of the teams.

As you begin to modify the Access Control system, note that the system has some project principal ACEs that should remain in place for correct system operation.

Server ACLs

The Server is the foundation for almost all ACL checks. Almost all objects in the system inherit these privileges. For this basic Team configuration, we will set the Server ACLs this way:

Type	Name	Read	Modify	Execute	Change
group	EC-administrator	A	A	A	A
group	Everyone	I	I	I	I

Custom Server Properties ACLs

An often overlooked ACL setting is Server Property Sheet. Allow the Everyone group Read permission on this object.

Type	Name	Read	Modify	Execute	Change
group	Everyone	A	I	I	I

System Object ACLs

There are 14 categories of System objects with ACLs.

1 - Administration – allow both designer groups Read privileges.

Type	Name	Read	Modify	Execute	Change
group	T1-designer	A	I	I	I
group	T2-designer	A	I	I	I

2 - Artifacts – allow the Everyone group Read privileges.

Type	Name	Read	Modify	Execute	Change
group	Everyone	A	I	I	I

3 - Directory – allow the Everyone group Read privileges.

Type	Name	Read	Modify	Execute	Change
group	Everyone	A	I	I	I

4 - Email Configurations – allow the Everyone group Read and Execute privileges.

Type	Name	Read	Modify	Execute	Change
group	Everyone	A	I	A	I

5 - Force Abort - no changes needed for this category.

6 - Logging - no changes needed for this category.

7 - Licensing - no changes needed for this category.

8 - Plugins - allow the Everyone group Read privilege. Generally, privileges set here will be inherited by all plugins.

Type	Name	Read	Modify	Execute	Change
group	Everyone	A	I	I	I

9 - Priority - no changes needed for this category.

10 - Projects - no changes needed for this category. Privileges for projects are now handled on a case by case basis.

11 - Repositories – allow the Everyone group Read privileges.

Type	Name	Read	Modify	Execute	Change
group	Everyone	A	I	I	I

12 - Resources - allow the Everyone group Read and Execute privileges.

Type	Name	Read	Modify	Execute	Change
group	Everyone	A	I	A	I

13 - Session - allow the Everyone group Execute privileges.

Type	Name	Read	Modify	Execute	Change
group	Everyone	I	I	A	I

14 - Workspaces - allow the Everyone group Read and Execute privileges.

Type	Name	Read	Modify	Execute	Change
group	Everyone	A	I	A	I

Plugin Project ACLs

Generally, you will want to allow the Everyone group Read privileges on every plugin you expect to use. (Each plugin has its own name, which is also the project name.) Because you may want to use many plugins, you might want to use a script.

Type	Name	Read	Modify	Execute	Change
group	Everyone	A	I	I	I

ACLs for Existing Projects

Next are a few Electric Cloud projects, and two sample projects that can be seen and used by their respective team members only.

Project: EC-Examples

These are nice examples and we want to grant Everyone the ability to see and run them.

Type	Name	Read	Modify	Execute	Change
group	Everyone	A	I	A	I

Project: Default

Inherited settings will allow the EC-administrator group full privileges.

Type	Name	Read	Modify	Execute	Change
group	Everyone	A	I	A	I

Project: EC-Utilities

Use caution when setting privileges on the EC-Utilities project. These are powerful procedures and some have their own ACL settings.

Remember the 'admin' user will always be able to use these utilities. Allowing the EC-administrator group full privileges accomplishes the same goal for users in that group. We allow the Everyone group Read only privileges. This is a unique project because it does not inherit ACLs from anywhere else (for example, Server or Projects).

Type	Name	Read	Modify	Execute	Change
group	EC-administrator	A	A	A	A
group	Everyone	A	I	I	I

Project: Electric Cloud

This project has one procedure and we want to grant everyone the ability to see it.

Type	Name	Read	Modify	Execute	Change
group	Everyone	A	I	I	I

ACLs for New Projects

Now we introduce the concept of ACL settings as they would be in a multiple team environment. For this example, we assume you have five projects being used by two teams in the following way.

- Project-A used by Team1
- Project-B used by Team1
- Project-C used by Team2
- Project-D used by Team2
- Project-E used by Team1 and Team2

Project-A

This project is visible and usable only by ElectricFlow users who are members of either the T1-user group or the T1-designer group. Notice that the designer group receives full permissions.

Type	Name	Read	Modify	Execute	Change
group	T1-designer	A	A	A	A
group	T1-user	A	I	A	I

Project-B

This project is the same as Project-A.

Type	Name	Read	Modify	Execute	Change
group	T1-designer	A	A	A	A
group	T1-user	A	I	A	I

Project-C

This project is visible and usable only by ElectricFlow users who are members of either the T2-user group or the T2-designer group. Notice that the designer group receives full permissions.

Type	Name	Read	Modify	Execute	Change
group	T2-designer	A	A	A	A
group	T2-user	A	I	A	I

Project-D

This project is the same as Project-C.

Type	Name	Read	Modify	Execute	Change
group	T2-designer	A	A	A	A
group	T2-user	A	I	A	I

Project-E

This project is a superset that includes both teams.

Type	Name	Read	Modify	Execute	Change
group	T1-designer	A	A	A	A
group	T1-user	A	I	A	I

Type	Name	Read	Modify	Execute	Change
group	T2-designer	A	A	A	A
group	T2-user	A	I	A	I

Enabling Same Team Subprocedures

To enable procedures in one project to use procedures in another project, one more step is necessary.

Add the project principals for both T1 projects to the T1-user group.

In this case, these are called: “project: Project-A” and “project: Project-B”.

Add the project principals for both T2 projects to the T2-user group.

In this case, these are called: “project: Project-C” and “project: Project-D”.

Optional: Restricting Resources and Workspaces by Team

Resource ACLs

Resource: local

Because each team will have dedicated resources, it is useful to keep one resource available for everyone to use.

Type	Name	Read	Modify	Execute	Change
group	Everyone	A	I	A	I

T1-resource

This resource is visible and usable only by ElectricFlow users who are members of either the T1-user group or the T1-designer group. This resource also needs Execute permission for the project itself to allow it to run from a schedule.

Type	Name	Read	Modify	Execute	Change
group	T1-designer	A	I	A	I
group	T1-user	A	I	A	I

T2-resource

This resource is visible and usable only by ElectricFlow users who are members of either the T2-user group or the T2-designer group. This resource also needs Execute permission for the project itself to allow it to run from a schedule.

Type	Name	Read	Modify	Execute	Change
group	EC-administrator	A	A	A	A
group	Everyone	I	I	I	I

Workspace ACLs

Workspace: default

Because there will be dedicated workspaces for each team, it is useful to keep one workspace available for Everyone to use.

Type	Name	Read	Modify	Execute	Change
group	Everyone	A	I	A	I

T1-workspace

This workspace is visible and usable only by ElectricFlow users who are members of either the T1-user group or the T1-designer group.

Type	Name	Read	Modify	Execute	Change
group	T1-designer	A	I	A	I
group	T1-user	A	I	A	I

T2-workspace

This workspace is visible and usable only by ElectricFlow users who are members of either the T2-user group or the T2-designer group.

Type	Name	Read	Modify	Execute	Change
group	T2-designer	A	I	A	I
group	T2-user	A	I	A	I

Artifact Management

[Artifact Objects](#)

[Access Control](#)

[Publishing Artifact Versions](#)

[Enable Compression](#)

[Include and Exclude Patterns](#)

[Retrieving Artifact Versions](#)

[Entering Filters During a Retrieve Operation](#)

[Dependent Artifact Versions](#)

[Artifact Cache](#)

[Cleaning Up Repositories and Caches](#)

Overview

During the course of the development process, project components or "outputs" are produced for consumption by other projects or the main project to create a whole entity. Ultimately, applications are assembled by combining these various outputs (dependencies) and packaging them together into a single deployable software application.

Outputs can be libraries, scripts, graphics, and so on. This output is an *artifact* that can be consumed by other projects.

Every stage in the application lifecycle produces and consumes artifacts. Artifacts are critical to the *build-test-deploy* process.

Using artifacts can:

- improve performance across builds
- provide better reusability of components
- improve cross-team collaboration with greater traceability

For example, instead of each developer repeatedly downloading third-party packages from external sources, these components can be published and versioned as an artifact. A developer then simply retrieves a specific artifact version from a local repository, guaranteeing a consistent package from build to build.

Artifact Objects

ElectricFlow has three objects to support Artifact Management functionality: *artifact*, *artifact version*, and *repository*. Similar to other ElectricFlow objects, each of these objects supports custom properties and access control.

Artifact

An *artifact* is a top-level object containing artifact versions, a name template for *published* artifact versions, artifact specific properties, and access control entries to specify privileges.

If an artifact is deleted, all published artifact versions within that artifact will be deleted also.

You can create artifacts in several ways:

- Use the New Artifact web page
- Use the `createArtifact` API command with the command-line tool (ectool) or an ec-perl script
- An artifact is created when you publish an artifact version if the artifact did not already exist, and the user issuing the publish command has permission to create artifacts. See the "Access Control" and "Publishing Artifact Versions" sections below for more information.

To create an artifact, you enter a Group Id and Artifact Key and together, these create an artifact name in the form `groupId:artifactKey`. After ElectricFlow creates the artifact name from your supplied Group Id and Artifact Key specifications, you cannot modify the name.

Note: Plan your artifact Group IDs and Artifact Keys carefully to prevent the extra work of deleting a name and beginning your specifications again. The use of a `groupId` and `artifactKey` allow two "namespaces" for added flexibility in organizing artifacts within your company or organization.

Examples

- If your development team uses GWT for web development and JUnit for unit testing, you may want to create artifacts `"ThirdParty:GWT"` and `"ThirdParty:JUnit"`.
- Your organization may have two different development teams (OS and Apps) who want to use artifacts. In building each product, both development groups want to publish and use an "SDK" with their product. But each SDK is different and unique. One solution would be to create one artifact named `"os:SDK"` and another artifact named `"apps:SDK"`. By using different group Id's, each development team is free to name their artifacts however they choose without worrying about name collisions with artifacts produced by the other team.

To reference an artifact, you can use the `groupId` and `artifactKey` combination or simply use the `artifactName`. The property path `/artifacts/<artifactName>` allows you to access any properties of a given artifact.

Artifact Version

An *artifact version* is a collection of 0 to N files that were published to an artifact repository. Artifact version metadata is stored in an `artifactVersion` object in the ElectricFlow server.

Tips for working with artifact versions:

- To group a collection of files (typically from build output in an ElectricFlow workspace) into an artifact version, specify the "from directory" containing those files.
- If you want to include files from multiple directories in your artifact, you need to specify a common root directory to include these directories when publishing your artifact.

- To restrict which files and subdirectories in the "from directory" to include in the artifact version, you can specify "include" and "exclude" patterns.
- Artifact versions are stored on an artifact repository server in either uncompressed `tar` archives or compressed `tar-gzip` archives.
- Artifact versions can be created using the `publishArtifactVersion` API, which includes creating the object in the ElectricFlow server and publishing the artifact version to the artifact repository.

Note: Interactively creating an artifact version using the automation platform web UI is not supported at this time.

- When you retrieve an artifact version, the original directory structure of the directory where files were published "from" is preserved.
- When you retrieve/publish artifacts, directories containing symlinks are always resolved; they are not preserved.

Artifact versions are referenced by *Group Id*, *Artifact Key* (or artifact name) and a version string.

- The artifact version's name is set based on the name template on the "owning" artifact. By default, the name template is in the form `groupId:artifactKey:version`.
- Each artifact version name must be unique within an ElectricFlow server installation.

The property path `/artifactVersions/<artifactVersionName>` allows access to any properties for a particular artifact version. Because you may not know the artifact version name (because of the name template), this path supports the form `<groupId>:<artifactKey>:<version>` as a substitute for the name of the artifact version.

All ElectricFlow API's that take an `artifactVersionName` argument also accept this alternate form as a substitute, similar to how other API's that accept a `jobId` argument accept the job name as a substitute.

Version Strings

A version string must be provided when publishing an artifact version.

For example:

`5.8.8-EN-55842` or `5.8.8.55842-EN` – In either form, the version string is interpreted as:

Major version number: 5

Minor version number: 8

Patch level number: 8

Qualifier: EN

Build or Job Id: 55842

You must provide separator punctuation. When interpreting (parsing) the first version string form, ElectricFlow interprets the text after the first hyphen as the build number if this string contains numeric characters only. Otherwise, the string is interpreted as the `qualifier`.

Version string examples and how ElectricFlow interprets them:

	Input (version string)	Major	Minor	Patch	Qualifier	Build Number
1	2.0.3-44834	2	0	3	<empty/null>	44834
2	3.8.4-RC1	3	8	4	RC1	0
3	3.8.4-RC1-36	3	8	4	RC1	36
4	3.8.4.36	3	8	4	<empty/null>	36
5	1-36	1	0	0	<empty/null>	36
6	\$_[jobId]-DE	\$_[jobId]	0	0	DE	0
7	3.8.4.36-RC1	3	8	4	RC1	36

In this table:

- Row 5 shows that you are not required to specify all three of `<major, minor, patch>` when publishing an artifact version if your organization's versioning conventions do not use all three fields. However, there is a caveat with this level of flexibility. You can create two artifact versions in the system with equivalent versions.
Version string "1-36" is equivalent to "1.0-36" but they are not the same so you can publish two artifact versions with these version strings successfully. When retrieving an artifact version, in some cases it could be unclear as to exactly which one you will get. For this reason, Electric Cloud recommends that you adopt a convention and abide by it for a particular artifact.
- Row 6 - A Job ID property was used as part of the version string. When expanded, the Job ID becomes a numeric string. Therefore, ElectricFlow interprets the Job ID as the "major version" number of the `major.minor.patch` string. DE becomes the qualifier, and no build number is set.

The artifact version object exposes components of the version string by using these intrinsic properties:

- **version** - the version component is the combination of `major.minor.patch-qualifier-buildNumber`. A "version" component does not need to include all of the following intrinsic properties. See the table above.
- **majorMinorPatch** - this is the `major.minor.patch` version component as specified in the version string. For example, if you specified a version string of "1-36, the `majorMinorPatch` component would be "1", not "1.0.0", which makes it possible for you to meet your naming conventions by reconstituting the version string a different way in the artifact version name template.
- **qualifier** - the qualifier component of the version .
- **buildNumber** - the build number component of the version.

The `artifactVersionState` property

Artifact versions have an `artifactVersionState` property whose value can be one of the following:

`publishing` - The artifact version is currently being published and is not available for retrieval.

`available` - The artifact version is available for retrieval if needed.

`unavailable` - The artifact version is not available for retrieval while in this state.

You can manipulate the state between available and unavailable on the Edit Artifact Version web page. This action can be useful if an artifact version seems to be corrupt in some way, but you want to investigate before potentially deleting it from the system.

By making the artifact version unavailable, retrievers can acquire an older (potentially more stable) artifact version while you look into the problem. If the artifact version is good, you can simply make it available again.

Repository

The artifact repository is a machine where artifact versions are stored. The repository server is configured to store artifact versions in a directory referred to as the repository backingstore.

By default, the backingstore is the `<datadir>/repository-data` directory in the repository installation. This default setting can be changed by running `ecconfigure --repositoryStorageDirectory` on the repository server. The repository server listens on port 8200 for HTTPS requests to publish and retrieve artifact versions.

Connection information is stored in the repository object on the ElectricFlow server.

You can create a repository objects in several ways:

- Use the New Repository web page in the ElectricFlow UI
- Use `createRepository` API
- Repository objects can be created automatically during the ElectricFlow installation.

When you are installing a repository *server*, the installer creates a corresponding repository *object* on your ElectricFlow server. If you are installing a repository *server* on the same machine as your ElectricFlow server, a repository *object* named "default" is automatically created. If you are installing a repository *server* on a different machine than your ElectricFlow server, the installer will prompt you for information it uses to create the corresponding repository *object*.

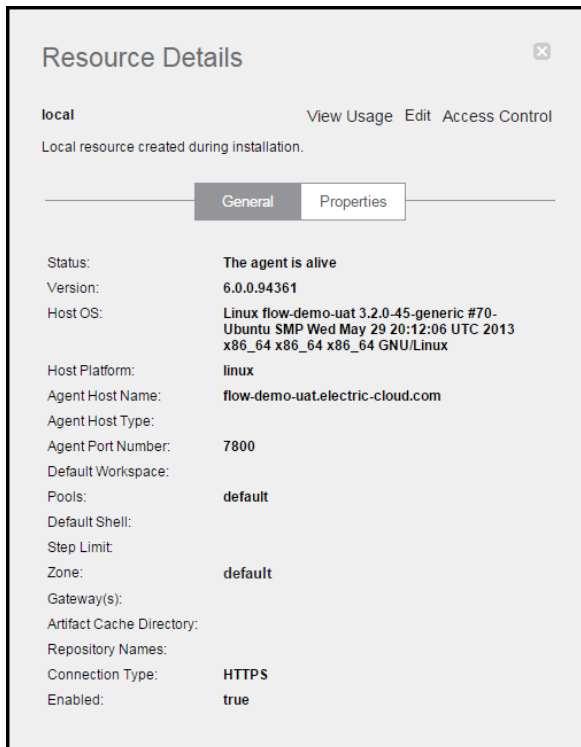
Note: When installing a repository server, Electric Cloud recommends installing an agent also to ease maintenance tasks such as clearing out stale artifact versions from the repository backingstore.

Click the Artifacts tab, then the Repositories subtab to see your list of created repositories.

Repositories					
All Repositories					
Add Repository					
Repository Name	Zone	URL	Enabled	Description	Actions
default	default	https://flow-demo-uat.electric-cloud.com:8200	☑	Default repository created during installation.	🗑
two	default	http://two:8201	☑	Created by the installer on two.	🗑
San Francisco	default	https://sf.electric-cloud.com:8200	☑	Repository for the San Francisco development team	🗑
Los Angeles	default	https://la.electric-cloud.com:8200	☑	Repository for the Los Angeles team	🗑

You can re-order the repository search order (for retrieving artifact versions) by using the mouse to grab and drag the icon in the left column to the row position you prefer, effectively re-ordering the repository list.

For distributed environments, where the preferred repository search order varies depending on which resource is performing the retrieval, you can specify a preference order on the Edit Resource panel. See the illustration below.

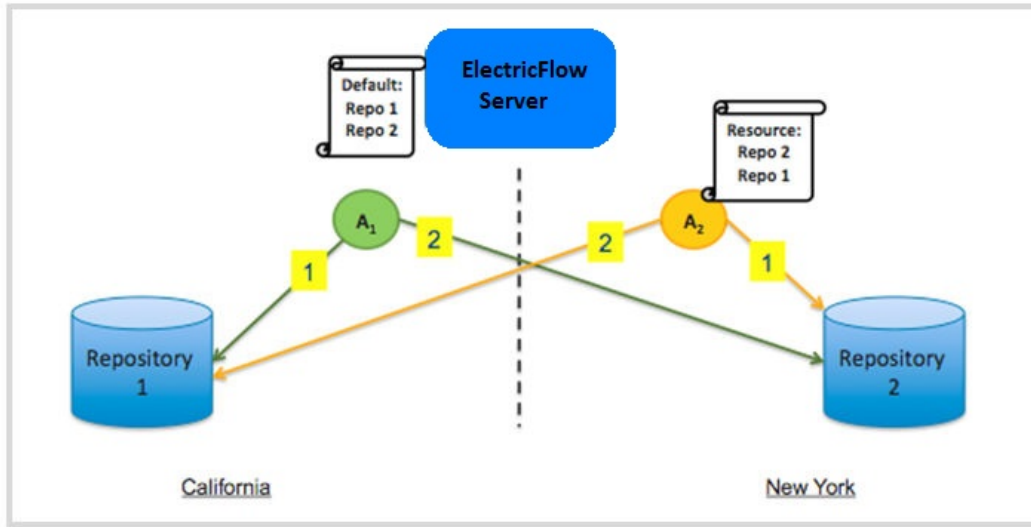


To go to the Edit Resource panel; from the Repositories page, select the Cloud tab, which opens to the Resources page, then select a resource name

Enter one repository name per line in the Repository Names field.

In the following example, a resource in California (A1) uses the "default" repository search order (that is, the order on the Repositories page). A1 searches for artifact versions by first requesting an artifact version from Repository 1. If the artifact version requested is not found, it will then request the artifact version from Repository 2.

The resource in New York (A2) has a specific search order specified where it searches for artifacts from Repository 2, first, then searches Repository 1.



Using multiple repositories

ElectricFlow supports multiple repositories, however you cannot share a repository between multiple ElectricFlow servers. Multiple repository servers may be required in your organization because of business or organizational reasons, government and compliance requirements, or for performance and data protection.

Examples:

Business or Organizational Structure

Within a company, two different development teams want to have artifact repositories containing artifacts and artifact versions specific to their product. A repository server is created for each team. The two teams publish artifact version to their respective repositories.

Performance

A company has multiple development sites. Instead of having a single repository server, which would create significant network performance issues from retrieving artifact versions during the build process, they create a repository at each development location and have artifact versions synchronized between the repository servers. In this way, when artifact versions are retrieved for use by a build process, they are retrieved from a local repository.

Data Protection and disaster recovery

A company has a single ElectricFlow server with two artifact repository server machines. The contents of the repository servers are synchronized and replicated between both machines. In the event repository 1 is unavailable during a retrieve, repository 2 is searched for the requested artifact retrieval.

Configuring the Artifact Repository Server for Amazon Simple Storage Service

You can configure the repository server to use Amazon Simple Storage Service (Amazon S3) as the backing store and connect to it. This section describes the changes to make in the `<DATA_`

DIRECTORY>/repository/server.properties and <DATA_DIRECTORY>/repository/wrapper.conf files.

Prerequisites

- Before starting this procedure, make sure the following have been created and are available on the AWS Management Console (<https://aws.amazon.com/console/>):
 - An Amazon S3 bucket for the backing store.
 - If you are using AWS Identity and Access Management (IAM) roles to connect to Amazon S3, the IAM role with a policy that grants all the `s3:*` access to the Amazon S3 bucket that you want to use with the repository server.

For example, the IAM role called `s3repositoryFullAccess` has a policy called `s3fullaccessAndAccessToInstallers` that grants all the `s3:*` access to the *backing store bucket* called `ec-test-repository-backingstore`.

Go to the Getting Started Guide (<http://docs.aws.amazon.com/AmazonS3/latest/gsg/GetStartedWithS3.html>) for further instructions.

Connecting to Amazon S3

Use one of these ways to access your AWS account:

- Use the Access Key ID and the Secret Access Key.

In `wrapper.conf`, set these properties:

```
wrapper.java.additional.400=-Daws.accessKeyId=YOUR_AWS_ACCESS_KEY_ID
```

```
wrapper.java.additional.401=-Daws.secretKey=YOUR_AWS_SECRET_KEY
```

- Use the IAM role from the previous section.

Go to the Getting Started Guide (<http://docs.aws.amazon.com/AmazonS3/latest/gsg/GetStartedWithS3.html>) for the detailed instructions.

Setting up the Artifact Repository

Follow these steps to set up the artifact repository for Amazon S3:

1. In the <DATA_DIRECTORY>/repository/server.properties file, set `REPOSITORY_BACKING_STORE` to use Amazon S3 by changing

```
REPOSITORY_BACKING_STORE=repository-data
```

to

```
REPOSITORY_BACKING_STORE=s3://<name of the backing store bucket>
```

2. Restart the artifact repository server using a command like this:

```
ubuntu@ip-10-101-214-28:~$ sudo /etc/init.d/commanderRepository restart
```

3. To test the `REPOSITORY_BACKING_STORE`, create two files in a `/tmp/<your username>` directory, publish the directory, and verify that it is published using commands like this:

```
ubuntu@ip-10-101-214-28:~$ cd /tmp
ubuntu@ip-10-101-214-28:/tmp$ mkdir <your username>
ubuntu@ip-10-101-214-28:/tmp$ cd <your username>
ubuntu@ip-10-101-214-28:/tmp/<your username>$ touch abc
ubuntu@ip-10-101-214-28:/tmp/<your username>$ ectool --server localhost login admin changeme
ubuntu@ip-10-101-214-28:/tmp/<your username>$ ectool publishArtifactVersion --version 1.0 --artifactName hello:world
```

4. Verify that the `/tmp/<your username>` is published to the Amazon S3 backing store.

The **repositoryDisabled** intrinsic property

The repository object has a `repositoryDisabled` property that dictates whether or not artifacts can be published or retrieved from a particular repository. If you take a repository offline for maintenance, you can disable the repository in the ElectricFlow server rather than shutting down the repository machine or the service. Steps that attempt to publish to this repository will fail, and steps that attempt to retrieve artifact versions will skip this repository during the search for available repositories.

Access Control

You can apply access control permissions to artifacts and artifact versions to control who can create an artifact versus who can publish an artifact version versus who can read or retrieve an artifact version.

Action	Required permission	Object
Create an artifact	Modify	artifacts system object
Read artifact metadata	Read	the relevant artifact
Modify an artifact	Modify	the relevant artifact
Delete an artifact, which deletes all artifact versions within the artifact	Modify	artifacts system object

Action	Required permission	Object
Publish an artifact version	Modify	the artifact into which a new version is being published
Read artifact version metadata (required for retrieval)	Read	the relevant artifact version
Modify an artifact version	Modify	the relevant artifact version
Delete an artifact version	Modify	the artifact that owns the artifact version

Note: The Execute privilege does not take part in artifact or artifact version access control.

"Out-of-the-box," an `<Everyone, modify>` access control entry is available that allows anyone to publish an artifact version to any artifact, and allows anyone to create an artifact. You might consider removing this access control entry and setting up tighter control for individual artifacts. See the EC-Security plugin for help with this effort.

For more information about access control for artifact management, see the [Access Control](#) Help topic.

The following scenarios provide ideas for how you might set up access control for artifact management.

Control who can create new artifacts and publish artifact versions

ElectricFlow administrator, Adam, received communication from development manager, Max, requesting that only he should be able to create artifacts in ElectricFlow -- developers of ElectricFlow procedures under him should only be able to create steps to publish artifact versions under pre-defined artifacts.

His reasoning is that it is too easy for a developer to introduce a typo during publish (which does auto-create the artifact if the user has the appropriate privilege and the artifact doesn't exist).

Adam accommodates the request by giving Max the modify privilege on the artifacts system object and removing the `<Everyone, modify>` access control entry. Max then creates artifacts and assigns modify privileges to those artifacts for his development group, effectively allowing them to publish artifact versions.

At run-time, a step belonging to some project actually performs the publish, so Max adds project principal access control entries for the relevant projects as well, providing modify privileges.

Control which projects can retrieve artifact versions at run-time

Acme Corporation's router product has a software stack consisting of OS, PlatformLibraries, and Apps. Each of these has a ElectricFlow project with several procedures that work together to build the relevant component. PlatformLibraries builds against some of the C header files and libraries produced by the OS build. Similarly, Apps builds against libraries produced by PlatformLibraries.

The lead developer for the OS component, creates a `Router:OSLib` artifact and gives the PlatformLibraries project principal read privileges so that project can retrieve any OSLib artifact version it needs to build PlatformLibraries. Similarly, Paul from the PlatformLibraries team creates a `Router:PlatformLib` artifact and gives the Apps project principal read privileges.

Group 1 publishes an artifact version and wants only Group 2 to be able to retrieve it

An organization wants to create artifacts for each of the third-party packages used by developers. Among the tools shared is the GWT package. The build manager chooses to create an artifact called `3rd-party-PKGS:GWT`. Because of licensing concerns, the build team must control or regulate who has access to the GWT and who can publish artifact versions used within the company's product.

In this case, the build team will publish a GWT artifact version and test it. During the test and approval cycles, only the build team has access to read and modify the artifact or publish new artifact versions. After a version has passed internal testing and approvals, the build team then grants read access to each of the developer groups.

Publishing Artifact Versions

An artifact version is published by using a procedure step or the `ectool /ec-perl` command-line interface. In most cases, using the Publish Artifact Version step is sufficient.

The following screen illustrates the parameters section for creating the Publish Artifact Version step:

Parameters

Artifact: Required

Version: Required

Repository: Required

Enable Compression? ☒

Follow Symbolic Links? ☒

From Directory:

+ Add Include Pattern

+ Add Exclude Pattern

+ Add Dependent Artifact Version

Enable Compression

When publishing an artifact, consider whether or not to enable compression. Using compression reduces transfer time during publish. However, compression also adds overhead when computing the compressed data. If files included in the artifact version are primarily text files or are another highly compressible file format, the benefit of reduced transfer time outweighs the cost of computing compressed data.

Artifact versions that contain installers, jars, audio, and video are almost certainly **not** compressible because these file types are already compressed, so the cost of compressing outweighs the (near zero) benefit of reducing transfer time. Another consideration: Artifact versions are stored in the same format as they are transferred, so highly compressible artifact versions use less space in the repository backingstore.

Include and Exclude Patterns

When publishing an artifact, you can choose which files to include by using include and exclude patterns. File patterns are expressed as relative paths under the From Directory. Pattern syntax and behavior is the same as Ant and uses the following wildcard characters:

? - matches a single character

* - matches any number of characters, but only at a single directory level

** - matches any number of directory levels

Examples:

Use *.jar to match any .jar file in the top-level directory.

Use */*.jar to match any .jar file in any child directory.

Use **/*.jar to match any .jar file at any level.

Retrieving Artifact Versions

An artifact version is retrieved using a procedure step, or the `retrieveArtifactVersion` API call, used by the `ectool/ec-perl` command-line tools. This API call returns the most current artifact version that meets your criteria, per the algorithm described below.

Typically, you specify a version number range. Artifact version ranges are specified using interval notation. Brackets `[]` indicate versions to include and parentheses `()` indicate versions to exclude.

For example, the following artifact versions are published:

```
foobar:test:3.0.1-3645
foobar:test:1.2.3-2139
foobar:test:2.0.0-4000
foobar:test:2.0.0-DE-3395
foobar:test:2.0.0-DE-2445
foobar:test:2.0.0-DE
foobar:test:3.0.0-3539
foobar:test:4.0.0-5584
```

You want to retrieve the most current artifact version for the `foobar:test` artifact. To do this, you do not need to provide a version or version range, but instead enter retrieve parameters in the Retrieve Artifact Version step page as follows:

Parameters

Artifact: foobar:test Required

Version:

☒ Latest
☐ Exact:
☐ Range:

Minimum:
☐ Inclusive?
Maximum:
☐ Inclusive?

Retrieve to directory:
☐

Overwrite:

update

Retrieved Artifact Location Property:

+Add Filter

The equivalent command using `ectool` is:

```
ectool retrieveArtifactVersions --artifactName foobar:test
```

and with `ec-perl`, the syntax is:

```
my $cmdr = newElectricCommander();
$cmdr->retrieveArtifactVersions({artifactName => "foobar:test"});
```

Version numbers for two artifact versions (Artifact Version 1 and Artifact Version 2) are compared in the server as follows:

Compare `major.minor.patch`. If they are equal,

Compare `qualifiers`. If they are equal,

Compare `buildNumbers`.

Note: This comparison algorithm does not take the time of publish into account. "Current" means the artifact version with the greatest version per the above algorithm.

So the above list of artifact versions would be sorted (most current to least as follows:

```
foobar:test:4.0.0-5584
foobar:test:3.0.1-3645
foobar:test:3.0.0-3539
foobar:test:2.0.0-DE-3395
foobar:test:2.0.0-DE-2445
foobar:test:2.0.0-DE
```



```
foobar:test:2.0.0-4000
```

```
foobar:test:1.2.3-2139
```

If you want to retrieve the latest 3.0.0 artifact version, set maximum version to 3.0.1 "exclusive".
The parameters entered in the retrieve step would look like this:

Setting this constraint ensures the entire "version string" is properly evaluated. In other words, you want to ensure all artifact versions with 3.0.0, including qualifiers and build numbers, are included in the evaluation.

The equivalent command using ectool is:

```
ectool retrieveArtifactVersions --artifactName foobar:test
--versionRange "(,3.0.1)"
```

And with ec-perl the syntax is:

```
my $cmdr = newElectricCommander();
$cmdr->retrieveArtifactVersions({artifactName => "foobar:test",
    versionRange => "(,3.0.1)"});
```

Note the use of parentheses to exclude version "3.0.1".

Generally, while in development, you will want to specify a minimum artifact version and take the latest version published, then when an artifact has reached a "released" state, indicate a specific version.

Entering Filters During a Retrieve Operation

At times, a company might want to distinguish between artifact versions not just by version range but also by some other property of the artifact version.

Examples are:

- Specifying a qualifier in the version string when publishing.
For example, you might want to publish an artifact version for your product's English or German variants.

You could publish the English version as `2.0.4-EN-55497` and the German version as `2.0.4-DE-55497`.

- Setting a custom property on the artifact version.
For example, after your QA team has tested an artifact version, they could show approval by setting a `qaApproval` custom property, including the name of the QA engineer.

For either case, filters can be applied when retrieving an artifact version. To retrieve the latest German version for the artifact, the parameters entered on the retrieve artifact custom step page my look like this:

The screenshot shows a 'Parameters' form with the following fields and values:

- Artifact:** `perl.pkg` (Required)
- Version:**
 - ☐ Latest
 - ☒ Exact:
 - ☐ Range:
 - Minimum: ☐ Inclusive?
 - Maximum: ☐ Inclusive?
- Retrieve to directory:** ☐ **Overwrite:** `update` (dropdown)
- Retrieved Artifact Location Property:** `/myJob/retrievedArtifactVersions/${assigned}`
- Filter(s):** `qualifier` (text), `equals` (dropdown), `DE` (text),
-

Because `DE` was used as the qualifier, ElectricFlow will retrieve the latest artifact version for the `perl:pkg` artifact where the qualifier equals `DE`.

Only the following filter operators can be applied to the intrinsic property version:

```

equals

greater than (greaterThan, for use in the API)

less than (lessThan, for use in the API)

greater than or equals (greaterOrEqual, for use in the API)

less than or equals (lessOrEqual, for use in the API)

```

"equals" is special because it does a string comparison on the version string of the artifact version, while the other operators compare the interpretation of the version string as described earlier.

All filter operators can be applied to version string components: `majorMinorPatch`, `qualifier`, or `buildNumber`. To find artifact versions with no qualifier, specify a filter on "qualifier" with operator "is not set" (or `isNull` in the API).

So for the QA approval case in a previous example, if a procedure is interested in an artifact version that was approved by QA, the retrieve step would specify a filter for `"qaApproval is set"`.

ectool does not currently support the filters argument, but you can write a Perl script as follows:

```

my $cmdr = new ElectricCommander();

$cmdr->retrieveArtifactVersions({artifactName => "perl:pkg",
    filters => {propertyName => "qaApprover",

```

```
operator => "isNotNull"}});
```

Dependent Artifact Versions

A published artifact version can be dependent on a list of artifact versions. These dependent artifact versions are retrieved when the primary artifact version is retrieved and they are specified with a query syntax when publishing the primary artifact version.

For example:

An artifact version for the ElectricFlow product could include dependent artifact versions for the core product, the SDK, and online Help:

- the latest `commander:Core` artifact version greater than or equal to version 3.5 (including 3.5)
- the most recent version of the `commander:SDK` artifacts
- the `commander:onlineHelp` artifact with a version greater than version 3.10 (excluding 3.10)

Because dependent artifact versions are evaluated during the retrieve process, specify them using the same syntax as you would for retrieving any artifact version, which is by including the group Id, the artifact key, and a specific artifact version or a version range.

Parameters for the "publish" custom step page would be similar to the following:

Parameters

Artifact: commander:PKG Required

Version: 4.0.0-\$(jobId) Required

Repository: default Required

Enable Compression? ☒

Follow Symbolic Links? ☒

From Directory:

Include Pattern(s):
Remove
Add Include Pattern

Exclude Pattern(s):
Remove
Add Exclude Pattern

Dependent Artifact Version(s):
commander:Core:[3.5,] Remove
commander:SDK:[1,] Remove
commander:onlineHelp:[3.10,] Remove
Add Dependent Artifact Version

A published artifact version could have zero files, that is, no files are published as part of the artifact version, but it still could have a list of dependent artifact versions. An empty artifact version might be useful as modular convenience container for retrieving a group of artifact versions in downstream processes.

Artifact Cache

Artifact versions are retrieved to an artifact cache directory before being consumed by a build process. Properly configured, the artifact cache directory can significantly improve efficiency as you work with artifacts.

The artifact cache directory is set in one of these ways:

- Enter information in the Artifact Cache Directory field on the Edit Resource page.
- Use the `modifyResource` API to set the `artifactCacheDirectory` intrinsic property.
- Set the `artifactCache` field in the `agent.conf` file.

Thus, each resource can have its own artifact cache location (directory). Alternatively, multiple resources and an entire site can share an artifact cache if the cache directory resides on a shared file server.

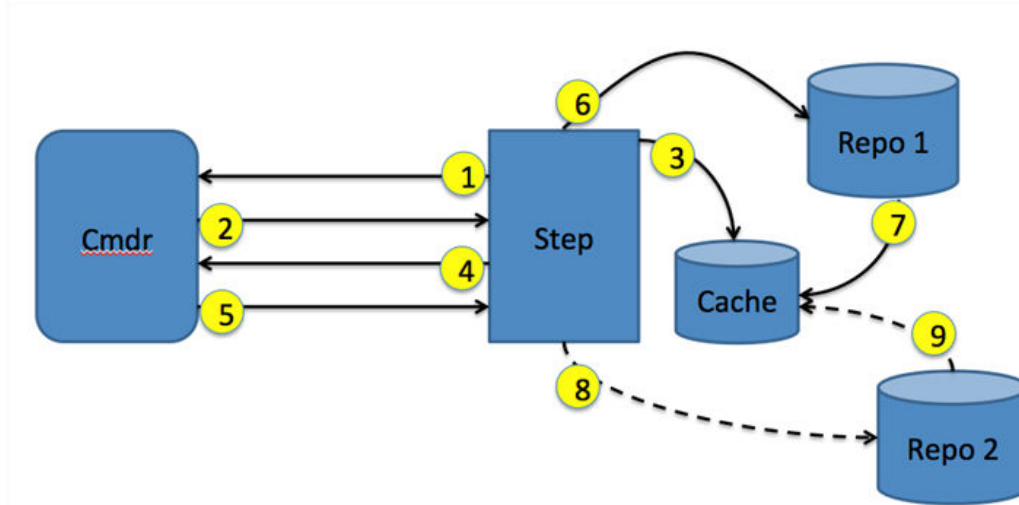
When an ElectricFlow step retrieves an artifact version, remember that often a version range or no version at all is specified. For example, to retrieve the latest ElectricFlow SDK artifact version, you could issue the following command:

```
ectool retrieveArtifactVersions --artifactName commander:SDK
```

No version is specified because we want the "latest" artifact version published.

In the following example, the step sends a request (1) to the ElectricFlow server.

- The ElectricFlow server replies (2) with a fully qualified artifact version name. In this case, it returns `commander:SDK:1.2.0.43552`.
- The step then looks in its artifact cache for the artifact version (3).
- If the artifact is not found in the artifact cache, a request is issued (4) to get a list of repositories to search to find the artifact version.
- The ElectricFlow server replies (5) with a list of repositories and the order in which to search them.
- The step requests the artifact version from the first (6) repository in the list.
- If the artifact version is found in the repository, the artifact version is retrieved to the cache (7).
- If the artifact version is not found, the step requests the artifact version from the next repository in the search order (8).
- If the artifact version is found in that repository, the artifact version is retrieved to the cache (9).
- Otherwise, if the artifact version is not found in any repository, an error is displayed.



In this example, if the requested artifact version was found in the artifact cache (step 3), the resource would not have communicated with either of the repository servers. Also, any potential network latencies would have been avoided from retrieving the artifact version from the repository server.

If you have a cache shared by multiple resources and steps that run on those resources need to use the same artifact version, the first step requesting an artifact version retrieves it from the repository to the cache. Subsequent job steps using the artifact version will have immediate access to the artifact version in the cache, eliminating the need for each step to retrieve its own copy of an artifact version.

Cleaning Up Repositories and Caches

Deleting an artifact version in the ElectricFlow server does not delete it in the repository backingstore, and it does not remove the artifact version from an artifact cache.

ectool and the ElectricFlow Perl module include a function for each of these tasks: `cleanupRepository` and `cleanupArtifactCache`. Both of these API calls are intended to be run on the machine containing the `cache/backingstore` directory that needs to be cleaned up.

The cleanup algorithm is as follows:

1. Review the directory tree looking for directories three-levels deep.
2. For each such directory, interpret the three path components as `<groupId, artifactKey, version>`.
3. Issue a `findObjects` request to the ElectricFlow server to see if an artifact version with that specification exists.
4. Delete all directories identified by `<groupId, artifactKey, version>` and not recognized by the ElectricFlow server.

To operate correctly, using the `cleanupRepository` API *requires* read access to all artifact version objects stored in ElectricFlow and therefore must be run in an "admin" session. Electric Cloud also recommends running `cleanupArtifactCache` in an "admin" session—if not, artifact versions that are simply not readable by the user invoking the cleanup will be deleted from the cache.

In addition, both functions fail if the server does not recognize any of the chosen artifact versions. Thus, if the cleanup function is called on a directory that is not a backingstore directory or an artifact cache, it will not delete anything.

The recommended approach for removing stale artifact versions from a backingstore is to use an ElectricFlow maintenance procedure that periodically runs on each repository server (which presumably has an agent install). Similarly, use a procedure with a broadcast step to run on each agent to clean up its cache. Note that the job step session does not necessarily have read privileges on all artifact versions, so it is important to login as a privileged user, then perform cleanup with those credentials.

Authenticating Users for LDAP and Active Directory

ElectricFlow uses account information from multiple sources. In most cases, the primary account information source is an external LDAP or Active Directory repository: both user and group information is retrieved from the repository. **Local** users and groups are defined within ElectricFlow.

Use the following LDAP information and examples to enter information in your LDAP template. If you are using the Active Directory template, you may still use the LDAP information as reference—the templates are similar with only a slight difference in the Active Directory template.

Configuring LDAP

A number of options need to be customized for any LDAP configuration. The following sample configuration shows how we have our own LDAP configuration set up. After the sample, see a list of properties with a description.

Sample LDAP Configuration

```
<bean id="LdapDirectoryProvider"
  class="com.electriccloud.security.ldap.
  LdapDirectoryProviderImpl">
  <property name="providerName" value="LDAP"/>
  <property name="url" value="ldap://dir.
    electric-cloud.com/dc=electric-cloud,dc=com"/>
  <property name="managerDn" value="uid=JohnDoe,ou=People,=
    electric-cloud,dc=com"/>
  <property name="managerPassword" value="****"/>
  <property name="userBase" value="ou=People"/>
  <property name="userSearchFilter" value="uid={0}"/>
  <property name="userNameAttribute" value="uid"/>
  <property name="fullUserNameAttribute" value="gecos"/>
  <property name="emailAttribute" value="mail"/>
  <property name="groupBase" value="ou=Group"/>
```

```

<property name="groupMemberFilter" value="(|(member={0})
    (memberUid={1}))"/>
<property name="groupMemberAttributes" value="member,
    memberUid"/>
<property name="groupSearchFilter" value="(|(
    groupOfNames) (objectClass=posixGroup))"/>
<property name="groupNameAttribute" value="cn"/>
</bean>

```

The following properties configure LDAP mapping:

emailAttribute - (optional) The attribute in a user record that contains the user's email address. If the attribute is not specified, the account name and domain name are concatenated to form an email address.

fullUserNameAttribute - (optional) The attribute in a user record that contains the user's full name (first and last) for display in the UI. If this attribute is not specified or the resulting value is empty, the user's account name is used instead.

managerDn - (optional) The DN of a user who has read-only access to LDAP user and group directories. If this property is not specified, the server attempts to connect as an unauthenticated user. Not all servers allow anonymous read-only access.

Note: This user does not need to be an admin user with modify privileges.

managerPassword - (optional) If the **managerDn** property is set, this password is used to authenticate the manager user.

providerName - This human readable name will be displayed in the user interface to identify users and groups that come from this provider.

userBase - This string is prepended to the **basedn** to construct the directory DN that contains user records.

userNameAttribute - The attribute in a user record that contains the user's account name.

userSearchFilter - This LDAP query is performed in the context of the user directory to search for a user by account name. The string "**r;{0}**" is replaced with the user's login ID. Typically, the query compares a user record attribute with the substituted user login ID.

url - The LDAP server URL is in the form **protocol://host:port/basedn**. Protocol is either **ldap** or **ldaps** for secure LDAP). The port is implied by the protocol, but can be overridden if it is not at the default location (389 for **ldap**, 636 for **ldaps**). The **basedn** is the path to the top level directory that contains users and groups at this site. This is typically the domain name where each part is listed with a **dc=** and separated by commas.

Note: Spaces in the **basedn** must be URL encoded (%20).

In addition to user information, the LDAP server can be queried for group information. This query is optional because the local group mechanism can refer to LDAP users and *local* users. However, the following elements can be used to tell the server how to map groups in LDAP

`groupBase` - (optional) This string is prepended to the `basedn` to construct the directory DN that contains group records.

`groupMemberAttributes` - (optional) A comma separated attribute names list that identifies a group member. Most LDAP configurations only specify a single value, but if there is a mixture of POSIX and LDAP style groups in the directory, multiple attributes might be required.

`groupMemberFilter` - (optional) This LDAP query is performed in the groups directory context to identify groups that contain a specific user as a member. There are two common forms of group record in LDAP directories: POSIX style groups where members are identified by account name, and `groupOfNames` or `uniqueGroupOfNames` records where members are identified by the full user DN. Both forms are supported, so the query is passed two parameters: "`r;{0}`" is replaced with the full user record DN, and "`r;{1}`" is replaced with the user's account name.

`groupNameAttribute` - (optional) The group record attribute that contains the name of the group.

`groupSearchFilter` - (optional) This LDAP query is performed in the context of the groups directory to enumerate group records.

Determining LDAP Mapping

A typical POSIX user record in LDAP looks similar the example below. To set up a mapping for this record, it is necessary to identify various record components. First, identify the path in the directory that contains user records. In this example, the build user has a distinguished name (dn) of `uid=build,ou=People,dc=mycompany,dc=com`. This name uniquely identifies the build user account and this path splits into three parts:

`base dn: dc=mycompany,dc=com`

`user base: ou=People`

`user element: uid=build`

The `baseDn` is the parent of the directory that contains users. This value should be combined with the protocol and server to form the URL. In this case, the URL is `ldaps://dir/dc=mycompany,dc=com`.

Next, the `userBase` is the portion of the path that identifies the directory containing all user account records. This value is used directly as the `userBase` configuration element.

The remaining portion identifies the user without the People directory: `uid=build`. The user name is replaced in this value with the string "`r;{0}`" to form the `userSearchFilter: uid={0}`. This query allows the server to search for a user's account name by looking for a record with a matching `uid` attribute.

The final mapping step is to identify user record attributes that hold the account name, full user name, and (optionally) the user's email address. In this example, the account name is `uid` (identified earlier), the full user name attribute is `gecos`, and there is no email attribute.

At this point, the server is able to authenticate a user, look up a user by name, and determine the user's full name. For many installations this is sufficient.

Sample LDAP User Record

```
# build, People, electric-cloud.com
dn: uid=jdoe, ou=People, dc=mycompany,dc=com
loginShell: /bin/bash
uidNumber: 508
gidNumber: 508
objectClass: account
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
uid: jdoe
gecos: John Doe
cn: John
homeDirectory: /net/filer/homes/build
```

Also, you can configure the server to look for **LDAP groups** that refer to user accounts. A typical group record is shown below. Like a user account, an LDAP group has a distinguished name with a `baseDn`, a group base, and a group element. In this case, the `basedn` is still `dc=mycompany,dc=com`. The `groupBase` configuration element is `ou=Group`, and the group name is `cn=build_users`.

The server needs to identify records in the directory that correspond to groups—it does this by applying the `groupMemberFilter` to the records in the `groupBase` directory. In this case, group records are members of the `posixAccount` object class, so the filter can be set to `objectClass=posixGroup`. To display a group by its name, the server needs to know which attribute represents the group name. In this case, set the `groupNameAttribute` to `cn`.

Finally, the server needs a filter to determine which accounts belong to the group and the attribute name that represents a single group member. Group membership can be identified in one of two ways. Either the members are listed by account name, or by their LDAP distinguished name. In this case, POSIX group membership is determined by account name, so set the `groupMemberAttributes` property to `memberUid`, and set the `groupMemberFilter` to `memberUid={1}`.

Sample LDAP Group Record

```
# build_users, Group, mycompany.com
dn: cn=build_users,ou=Group,dc=mycompany,dc=com
objectClass: posixGroup
objectClass: top
gidNumber: 100
memberUid: jdoe
memberUid: mary
cn: build_users
```

Sample Active Directory Configuration File

The following XML file defines parameters needed to connect to an Active Directory server and the query to use for looking up user information.

```
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:
xmlns:tx
xmlns:aop
```

```
xsi:schemaLocation
http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-2.0.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-2.0.xsd" default-lazy-
init="true"

    <bean id="ADDirectoryProvider"
        class="com.electriccloud.security.ldap.ActiveDirectoryProviderImpl">
        <property name="providerName" value="ActiveDirectory" />
        <!-- START OF CUSTOMIZATIONS -->
        <!-- URL to AD server of the form: "r;ldap://host:port/base_dn" -->
        <property name="url" value="ldap://server:389/dc=company,dc=com" />
        <!-- Required credentials for an account that has read-only access to the server
        -->
        <property name="managerDn"
            value="cn=myuser,cn=Users,dc=company,dc=com" />
        <property name="managerPassword" value="mypw" />
        <property name="userBase" value="cn=Users" />
        <property name="userSearchFilter" value="(&(sAMAccountName={0})
            (objectClass=user))" />
        <property name="userNameAttribute" value="sAMAccountName" />
        <property name="fullUserNameAttribute" value="name" />
        <!-- Optional group configuration. This is only needed if you intend -->
        <!-- to manage groups externally via AD. -->
        <!-- <property name="groupBase" value="value=""/>-->
        <!-- <property name="groupMemberFilter" value="member={0}"/> -->
        <!-- <property name="groupMemberAttributes" value="member"/>-->
        <!-- <property name="groupSearchFilter" value="(objectClass=group)"/>-->
        <!-- <property name="groupNameAttribute" value="cn"/>-->
        <!-- Max number of results to get back in one query. This must be -->
        <!-- no greater than the AD server's setting, which is typically 1000. -->
        <property name="pageSize" value="500" />
        <!-- END OF CUSTOMIZATIONS -->
    </bean>
</beans>
```

Change Tracking

Change Tracking is designed to track every change between every state of non-runtime ElectricFlow objects and to allow you to export or revert to any previous state of these objects. ElectricFlow tracks the changes to tracked objects including applications, procedures, workflows, workspaces, resources, and project-owned components such as library components and records a *Change History* of the historical states of the system and the changes between them. The change history has a copy of every state in which every non-runtime object has been, the object's current state, and indexing records for searching through the database. The tracked objects are not affected when ElectricFlow executes an object that is usually created or modified at execution time, such as a workflow or process.

Change tracking allows you to do the following:

- When debugging a failed job or looking for more information about a component in an ElectricFlow, see the Change History for the changes relevant to that object.
- When searching for specific change history records, filter the records by time frame, change type, entity type, or developer.
- Revert changes to an object or to an objects and its children.
- When you want to determine the differences between objects, export them at various levels in the object hierarchy.

In ElectricFlow, you can use Change Tracking in all aspects of the ElectricFlow end-to-end solution:

- In build-test automation, track the Change History of artifacts, resources, and ACLs.
- In deployment automation, track the Change History of components, environment tiers, and process steps (application or component). You also use Change Tracking with snapshots to make it easier to deploy reliable and repeatable software for Continuous Delivery.
- In pipeline automation, track the Change History of stages and tasks.

More about application, deploy, and run:

As you use ElectricFlow, remember that these terms have different meanings within ElectricFlow *and* outside of ElectricFlow when you deploy your software or application:

Term	Within ElectricFlow	Outside of ElectricFlow
Application	The application that you design and run (deploy) to produce your software for continuous delivery across different pipelines.	The software, system or application that you build, test, install, implement, release, and deploy using ElectricFlow. This is the end product of using ElectricFlow.
Deploy	Running the application that you designed in ElectricFlow. The end product is your software, system, or application. Deploy is a synonym of run in ElectricFlow.	All the processes or actions to develop and run your software in its environment, including building, testing, implementing, installing, configuring, making changes, and releasing.
Run	Running the application that you designed. The end product is your software, system, or application. <i>Run</i> is a synonym of <i>deploy</i> in ElectricFlow.	All the processes or actions to use software in its environment, including implementing, installing, configuring, debugging, troubleshooting, and releasing.

Configuring Change Tracking

Change Tracking must be enabled when ElectricFlow starts for your system to track changes and record the Change History.

By default, Change Tracking is enabled for projects created in ElectricFlow 5.3 or later and for projects created before upgrading to ElectricFlow 5.3. When you import a project created before ElectricFlow 5.3, Change Tracking is disabled for the project.

Enabling Change Tracking Globally

When installing ElectricFlow:

1. Add this line to the `database.properties` file:

`COMMANDER_DB_AUDITING_ENABLED=true`
2. Restart the ElectricFlow server.

Enabling Change Tracking on a Per-Project Basis

IMPORTANT: Unless the `allowEnablingDisablingChangeTrackingProjectLevel` setting has been altered, only admin users are allowed to perform this task. The default setting is `adminOnly`. If the setting is changed to `anyoneWithAccess`, any user with access to the project can enable or disable Change Tracking for the project

You can enable Change Tracking one of the following ways:

In the ElectricFlow Platform UI

Change Tracking is enabled when the **Enable Change Tracking** check box is selected.

The screenshot shows a 'New Project' form with the following fields and controls:

- Name:** A text input field with a cursor.
- Enable Change Tracking:** A checked checkbox.
- Description:** A large text area.
- Default Resource:** A text input field with a 'Browse' button to its right.
- Default Workspace:** A text input field with a 'Browse' button to its right.

To disable Change Tracking, click the **Enable Change Tracking** check box to clear it, and click **OK**.

Using ectool

- Enter `ectool modifyProject <projectName> --tracked true` to enable Change Tracking.
- Enter `ectool modifyProject <projectName> --tracked false` to disable Change Tracking.

Using ec-perl:

- Enter `$cmdr->modifyProject(<projectName>, {tracked => true});` to enable Change Tracking.
- Enter `$cmdr->modifyProject(<projectName>, {tracked => false});` to disable Change Tracking.

For properties that are frequently updated always to be a numerical value, it is possible to suppress Change Tracking of numerical-value updates by using one of these commands:

ectool: `ectool modifyProperty <projectName> -- path <propertyPath> -- counter true`

or

ec-perl: `$cmdr-> modifyProperty (<projectName>, {path => <propertyPath>, counter => true});`

When the `counter` flag is set for a property, Change Tracking does not track changes to the property if the only change was a change in the property value from one numeric value to another. All other forms of changes to the properties that have this flag set are tracked normally.

CAUTION: Reverting an object that owns a counter property to a previous state will revert the value of the counter property to its value at the previous time that a change to this property was tracked—typically this is when its value is initialized. This may not be the desired behavior, so you may need to manually set a counter property if it is reverted,.

Upgrading to ElectricFlow 6.x

Change Tracking is enabled when you upgrade to ElectricFlow 6.x. This can significantly increase the time it takes to complete the upgrade.

If you want to upgrade with Change Tracking disabled, add this line to the `database.properties` file before starting the upgrade:

```
COMMANDER_DB_AUDITING_ENABLED=false
```

Customizing the Change History Page

The performance of the Change Tracking feature is affected by number of records in the Change History as well as the number of entries being tracked. For example, a page showing 5000 entries may be slow to load and update and does not provide much useful information.

This ElectricFlow server uses the lowest of the following limits to determine the maximum amount of records to display in the Change History page:

- Maximum amount of records on the Change History page

To change the maximum number of records in the Change History page:

1. Set the `CHANGE_TRACKING_HARD_MAX_RECORDS` parameter in the `wrapper.conf` file to a new value.

The default value is 1000.

2. Restart the ElectricFlow server.

- Maximum number of records retrieved

Set the `TrackingMax Records` server setting to a new value not exceeding the `CHANGE_TRACKING_HARD_MAX_RECORDS` parameter in the `wrapper.conf` file.

To set `TrackingMaxRecords`, do one of the following:

Change the value in the UI. See the Server Settings page in the automation platform UI.

Use `ectool` to change the value. For example, enter the following command to limit the number of records retrieved to 100:

```
ectool setProperty /server/settings/changeTrackingMaxRecords --value 100
```

Viewing the Change History for Artifacts, Jobs, Projects, and Workflows

When troubleshooting why a job failed, you can view the Change History for artifacts, jobs, projects, and workflows in the automation platform.

- [Artifacts](#) on page 953
- [Jobs](#) on page 954
- [Projects](#) on page 955
- [Workflows](#) on page 956

Artifacts

Starting from the Home page:

1. Go to the Artifacts tab.
2. Choose an artifact.

3. Click the **Track Changes** button.

Example:

Artifacts

4 Results

New Search | Create Artifact

Name	Group Id	Artifact Key	Description	Actions
com.ec.myapp	com.ec	myapp		<div><div></div><div></div><div></div></div>
com.mycompany.heatclinic.config	com.mycompany.heatclinic	config		<div><div></div><div></div><div></div></div>
com.mycompany.heatclinic.warfile	com.mycompany.heatclinic	warfile		<div><div></div><div></div><div></div></div>
com.mycompany.heatclinic.warfileWildfire	com.mycompany.heatclinic	warfileWildfire		<div><div></div><div></div><div></div></div>

Records per page: 20

1 thru 4 of 4

The Change History for the selected artifact opens.

The default time increment is **Past 60 Minutes**.

Example:

Change History

Change History for com.ec:myapp

Last changes

7/10/15 - 1:02 PM

Now - 7/10/15 - 6:49 PM

Most Recent

View All Changes

Objects

Acl (2)

Property Sheet (1)

Artifact (1)

Changes

Created (4)

Changed by...

Admin (4)

When	What	Name	By...	Change	Path
1 Jul 10, 2015 1:02 PM Pacific...	artifact	com.ec:...	admin	created	

Jobs

Starting from the Home page:

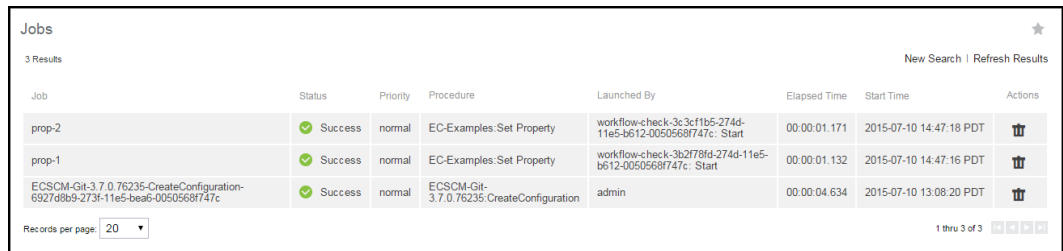
1. To go to the Job Details page, do one of the following:
- Use the Jobs tab.
 - Use the Jobs Quick View list.

2. If you use the Jobs tab, follow these steps:

a. Click the Jobs tab.

The Jobs page opens.

Example:



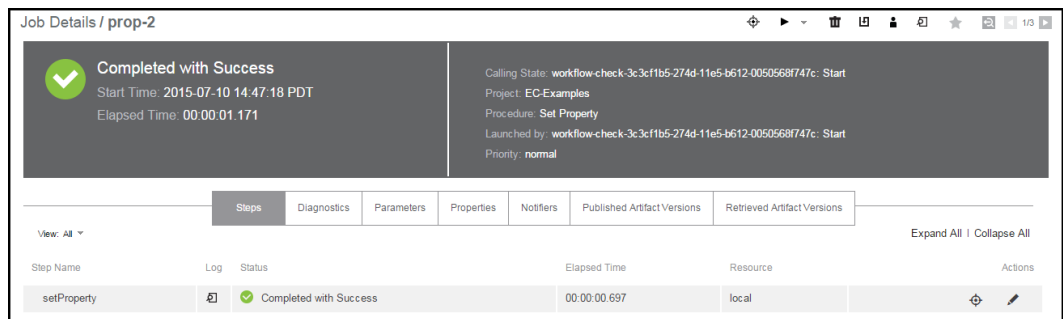
The screenshot shows the 'Jobs' page with a table of results. The table has columns for Job, Status, Priority, Procedure, Launched By, Elapsed Time, Start Time, and Actions. There are three rows of job results, all with a 'Success' status. The first row is for 'prop-2', the second for 'prop-1', and the third for a long ID. The 'Launched By' column shows 'admin' for the third row. The 'Actions' column contains a trash icon for each row. At the bottom, there is a 'Records per page' dropdown set to 20 and a pagination indicator '1 thru 3 of 3'.

Job	Status	Priority	Procedure	Launched By	Elapsed Time	Start Time	Actions
prop-2	Success	normal	EC-Examples: Set Property	workflow-check-3c3cf1b5-274d-11e5-b612-0050568f747c: Start	00:00:01.171	2015-07-10 14:47:18 PDT	🗑️
prop-1	Success	normal	EC-Examples: Set Property	workflow-check-3b2f78fd-274d-11e5-b612-0050568f747c: Start	00:00:01.132	2015-07-10 14:47:16 PDT	🗑️
ECSCM-Git-3.7.0.76235-CreateConfiguration-6927d8b9-273f-11e5-bea6-0050568f747c	Success	normal	ECSCM-Git-3.7.0.76235: CreateConfiguration	admin	00:00:04.634	2015-07-10 13:08:20 PDT	🗑️

b. Click a job name to select a job.

The Job Details page opens.

Example:



The screenshot shows the 'Job Details / prop-2' page. It features a large green checkmark icon and the text 'Completed with Success'. Below this, it shows 'Start Time: 2015-07-10 14:47:18 PDT' and 'Elapsed Time: 00:00:01.171'. To the right, there is a summary of the job: 'Calling State: workflow-check-3c3cf1b5-274d-11e5-b612-0050568f747c: Start', 'Project: EC-Examples', 'Procedure: Set Property', 'Launched by: workflow-check-3c3cf1b5-274d-11e5-b612-0050568f747c: Start', and 'Priority: normal'. Below the summary, there are tabs for 'Steps', 'Diagnostics', 'Parameters', 'Properties', 'Notifiers', 'Published Artifact Versions', and 'Retrieved Artifact Versions'. The 'Steps' tab is selected, showing a table with columns for Step Name, Log, Status, Elapsed Time, Resource, and Actions. The table has one row: 'setProperty' with a status of 'Completed with Success' and an elapsed time of '00:00:00.697'. The resource is 'local'. There are icons for expanding/collapsing and editing the step.

Step Name	Log	Status	Elapsed Time	Resource	Actions
setProperty	📄	Completed with Success	00:00:00.697	local	🔍 ✎️

3. If you use the Jobs Quick View list, click a job name to select a job.

The Job Details page opens.

4. Choose a job or job step.

5. Click **Track Changes** for the job or job step.

The Change History for the job or job step opens.

The default time increment is **Past 60 Minutes**.

Projects

Starting from the Home page:

1. Go to the Projects tab.

2. Choose a project.

- Click the **Track Changes** button.

The Change History for the project opens.

Example:

Projects

11 Results

All Projects | Create Project | New Search

Project	Description	Impersonation Credential	Create Date	Actions			
Branch_1_2_9	Branch_1_2_9		2015-07-10 15:51:41 PDT				
Default	Default project created during installation.		2015-07-10 12:56:39 PDT				
DemoLibrary	Library of procedures used by the ElectricFlow Demo System.		2015-06-12 00:56:23 PDT				
DeployObjects-1.7.8			2015-07-10 13:02:26 PDT				
EC-Examples	This project contains templates for procedures that perform basic tasks within ElectricCommander.		2015-07-10 12:56:40 PDT				
EC-Tutorials-1.0.0.69904	This project contains small pieces of code which show how to use features in ElectricCommander.		2015-07-10 12:57:51 PDT				
EC-Utilities	This project contains procedures that perform basic tasks within ElectricCommander. Also, the procedures can be used as examples that can be copied and modified in another project.		2015-07-10 12:56:50 PDT				
Electric Cloud	Electric Cloud Procedures		2015-07-10 12:57:18 PDT				
beanstalk	AWS procedures to manipulate Amazon resources for Elastic Beanstalk operations: <ul style="list-style-type: none">S3 storage for file deploymentsElastic Beanstalk EnvironmentsElastic Beanstalk Applications		2015-01-06 06:47:18 PST				
ec-library	Library of Electric Cloud utility procedures.		2015-01-07 05:55:23 PST				
jpetstore			2015-02-19 05:06:40 PST				

Records per page: 20

1 thru 11 of 11

Records per page: 20

1 thru 11 of 11

Workflows

Starting from the Home page:

- Go to the Workflows tab.
- Choose a workflow.
- Click **Track Changes**.

The Change History for the workflow opens.

Modifying the Change History

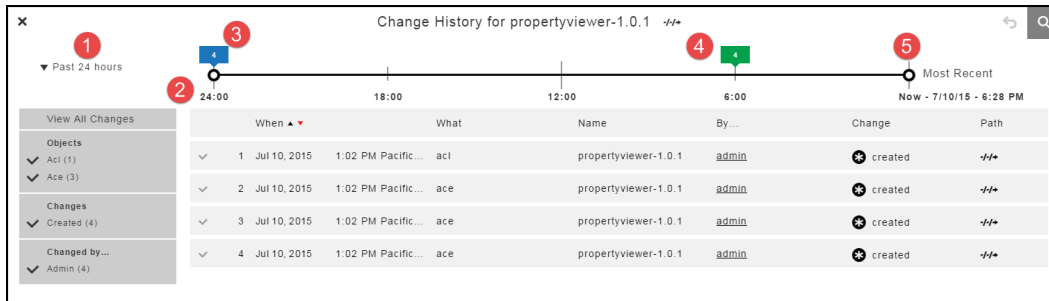
You can modify the information that appears in the Change History with these settings:

- Time increment—Go to [Change History Time Line](#) on page 956.
- Time line—Go to [Change History Time Line](#) on page 956.
- Filters—Go to [Change History Filters](#) on page 962.

Change History Time Line

The time line is at the top of the Change History page.

This example shows the Change History for a property called *propertyviewer-1.0.1*.



1	<p>Time increment.</p> <p>The default is Last Changes.</p> <p>Click the down arrow to select another time increment.</p>
2	<p>The system automatically calculates the minutes, hours, and days since the last successful run.</p> <p>In the example, the last successful run occurred 24 hours ago. The time line is divided into four 6 hour subdivisions.</p>
3	<p>Total number of changes in the selected time increment.</p>
4	<p>The number of changes that occurred between 6 hours ago and now is 4.</p> <p>When you click the change number, the Change History is updated and shows only those changes.</p>
5	<p>Drag the start and end time markers to view specific changes.</p>

Default Settings

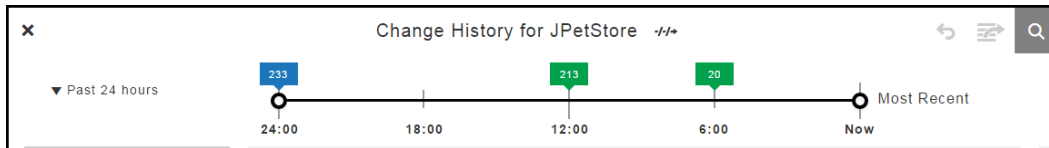
The default time increment is **Last Change**.

The entire time line is displayed, and all the changes are in the list below the time line.

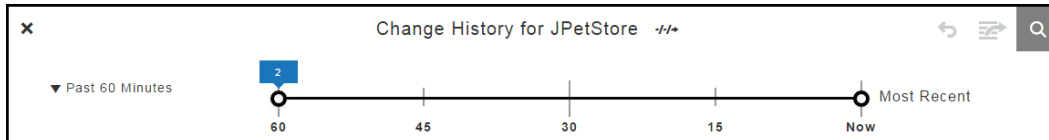
Number of Changes

The time line shows the number of changes throughout the time increment. In the following example:

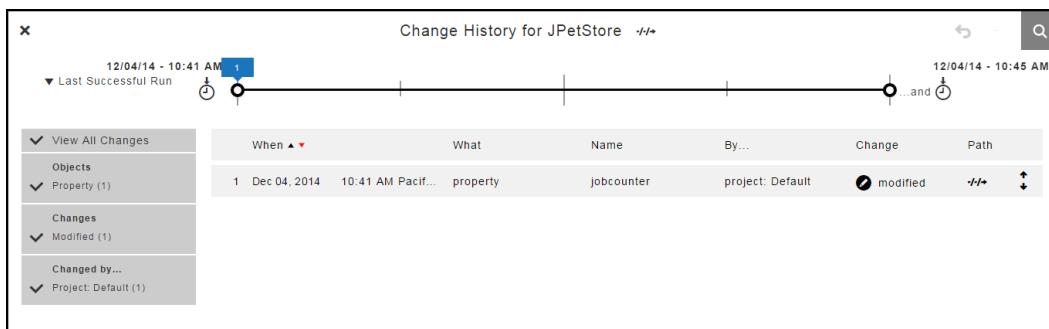
- There have been 233 changes in the last 24 hours.
- There have been 213 changes in the last 12 hours.
- There have been 20 changes in the last 6 hours.



When you change the time increment, there have been two changes in the previous 60 minutes.



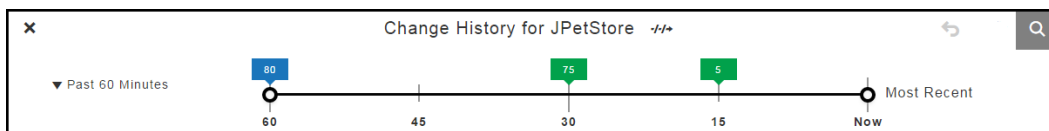
Time Increment



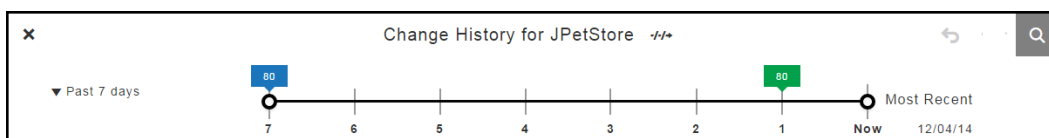
The system automatically determines how the time line is divided for the selected time increment.

When the range is changed to **Past 60 Minutes**, the time line changes:

- The start time is 60 minutes from **Now**.
- The end time is when the **Most Recent** change occurred (**Now**).
- The time line has four divisions.



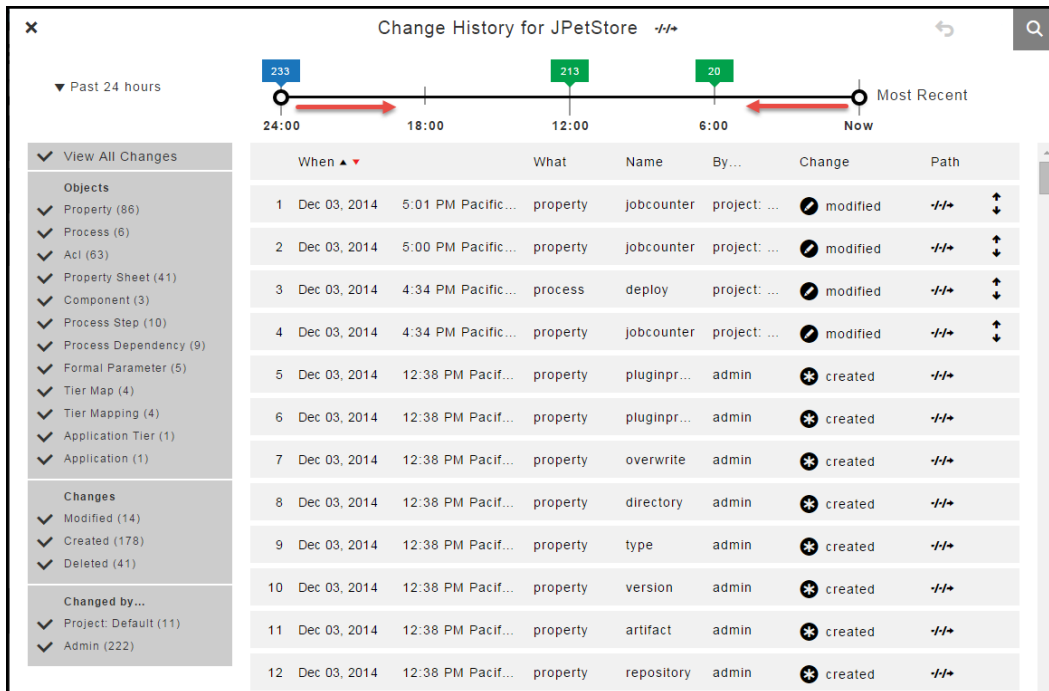
If the increment is **Past 7 Days**, the time line has seven one-day divisions.



Moving the Start and End Times Manually

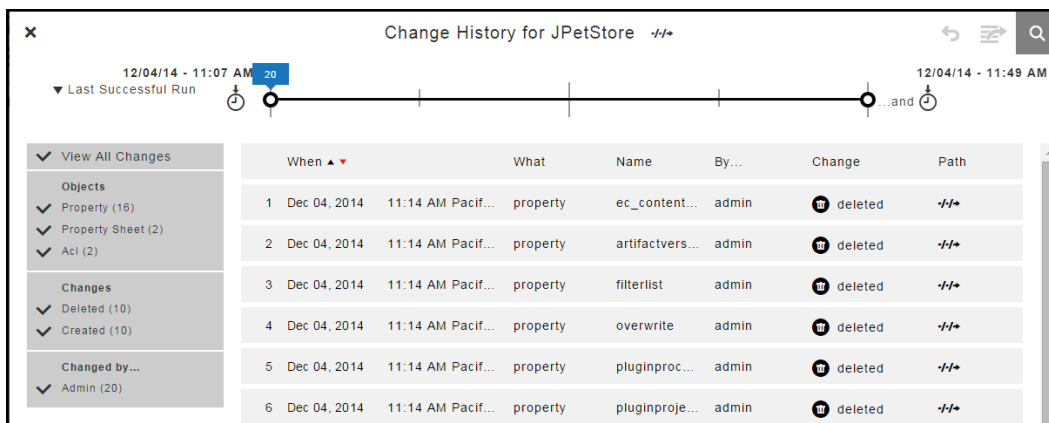
You can manually move the start and end times on the time line.

When you move the start time to 18:00 and the end time to 6:00, the list of objects in the change history changes.



Setting Custom Time Increments

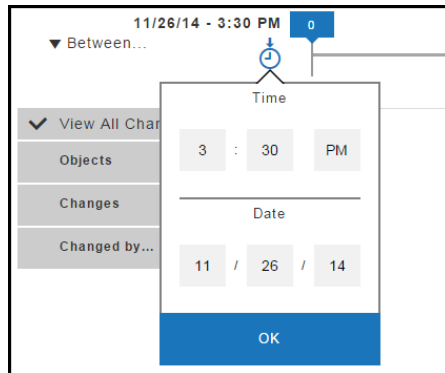
Example:



1. Select **Between**.

A drop down dialog box opens.

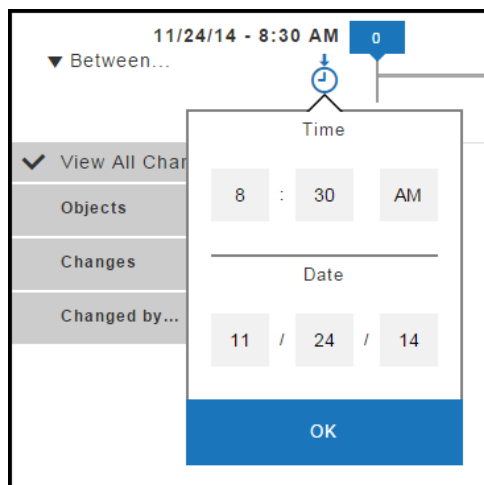
Example:



2. Select the time and date for the start of the time line.

The default settings are **3:30 PM** and eight days before the current date.

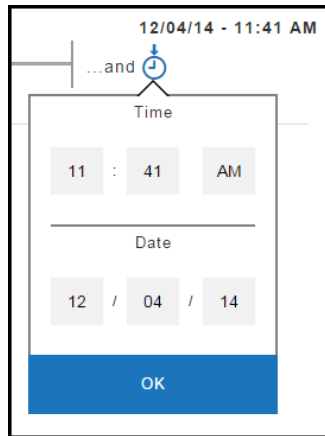
Example:



3. Click **OK**.

A drop down dialog box opens at the other end of the time line.

Example:

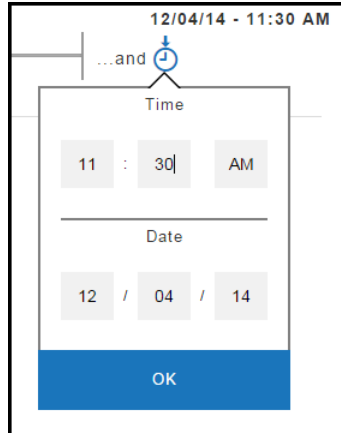


The screenshot shows a dialog box titled "Time" and "Date" with a blue "OK" button at the bottom. The "Time" section displays "11 : 41 AM" and the "Date" section displays "12 / 04 / 14". Above the dialog box, a timeline icon is visible with the text "12/04/14 - 11:41 AM" and "...and" to its left.

4. Select the time and date for the end of the time line.

The defaults are **3:30 PM** and the current date.

Example:



The screenshot shows a dialog box titled "Time" and "Date" with a blue "OK" button at the bottom. The "Time" section displays "11 : 30 AM" and the "Date" section displays "12 / 04 / 14". Above the dialog box, a timeline icon is visible with the text "12/04/14 - 11:30 AM" and "...and" to its left.

5. Click **OK**.

The time line changes to show only the changes from the start and end times and dates that you selected.

Example:

Change History for JPetStore

11/24/14 - 8:30 AM 100 100 12/04/14 - 11:30 AM

Between... and...

View All Changes	When ▲ ▼	What	Name	By...	Change	Path
Objects						
✓ Acl (25)	1 Dec 04, 2014 10:28 AM Pacific...	acl	jpetstore	admin	* created	-/+/
✓ Property Sheet (16)	2 Dec 04, 2014 10:28 AM Pacific...	acl	jpetstore	admin	* created	-/+/
✓ Application (1)	3 Dec 04, 2014 10:28 AM Pacific...	propertySh...	jpetstore	admin	* created	-/+/
✓ Property (41)	4 Dec 04, 2014 10:28 AM Pacific...	propertySh...	ec_deploy	admin	* created	-/+/
✓ Application Tier (1)	5 Dec 04, 2014 10:28 AM Pacific...	application	jpetstore	admin	* created	-/+/
✓ Component (1)	6 Dec 04, 2014 10:28 AM Pacific...	property	ec_notifier...	admin	* created	-/+/
✓ Process (2)	7 Dec 04, 2014 10:28 AM Pacific...	property	ec_deploy	admin	* created	-/+/
✓ Process Step (2)	8 Dec 04, 2014 10:28 AM Pacific...	acl	ec_deploy	admin	* created	-/+/
✓ Formal Parameter (3)	9 Dec 04, 2014 10:28 AM Pacific...	acl	apptier	admin	* created	-/+/
✓ Tier Map (4)	10 Dec 04, 2014 10:28 AM Pacific...	propertySh...	apptier	admin	* created	-/+/
✓ Tier Mapping (4)	11 Dec 04, 2014 10:28 AM Pacific...	acl	apptier	admin	* created	-/+/
Changes						
✓ Created (86)						
✓ Modified (4)						
✓ Deleted (10)						
Changed by...						
✓ Admin (95)						
✓ Project: Default (5)						

Change History Filters

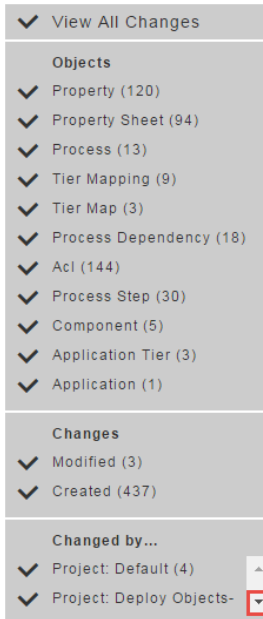
You can use filters to view changes to specific objects, the types of changes, and the users how made those changes.

Instead of selecting **View All Changes**, you can select specific objects, such as only properties, processes, property sheets, process steps, and process dependencies that have been modified by the Project:Default and Admin users.

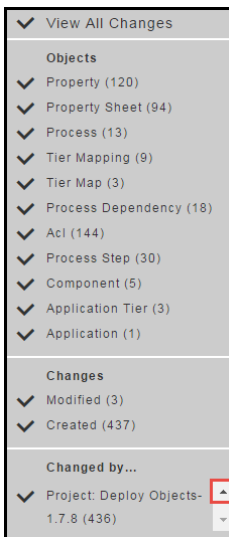
View All Changes	When ▲ ▼	What	Name	By...	Change	Path
Objects						
✓ Property (43)	1 Dec 03, 2014 4:34 PM Pacific...	property	jobcounter	project: ...	modified	↑ ↓
✓ Process (2)	2 Dec 03, 2014 4:34 PM Pacific...	process	deploy	project: ...	modified	↑ ↓
✓ Property Sheet (20)	3 Dec 03, 2014 11:35 AM Pacific...	process	deploy	admin	modified	↑ ↓
Acl (27)						
Component (1)						
✓ Process Step (6)						
✓ Process Dependency (8)						
Changes						
✓ Modified (6)						
Created (60)						
Deleted (41)						
Changed by...						
✓ Project: Default (2)						
✓ Admin (105)						

When the list of filter criteria is long, not all of the criteria may appear in the filter list. To see all of the criteria, use the up or down arrows to see all the options.

This list does not show all of the users. Use the up and down arrows to see all four of the users.



Click the down arrow to see the other users.



Searching the Change History

Follow these steps to start a search in the Change History.

You can start a Change History search from most pages in the ElectricFlow UI.

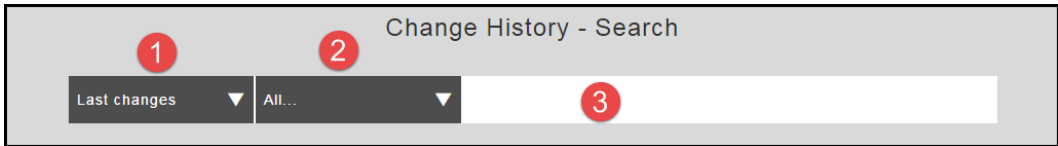
1. Click the **Search** button or click **Change History**.

Example:



The **Change History - Search** dialog box opens.

Example:



1	<p>Time range field.</p> <p>Click the down arrow to open the drop-down list of start times.</p> <p>The end time is the current time.</p>
2	<p>Objects field.</p> <p>Click the down arrow to open the drop-down list of objects to include in the search. You can select All or specific objects.</p> <p>By default, seven of the most commonly tracked objects are selected.</p>
3	<p>Search criteria.</p> <p>After you type, the system starts searching for objects based on the time range and objects that you selected.</p> <p>The search results are in the Change History.</p>

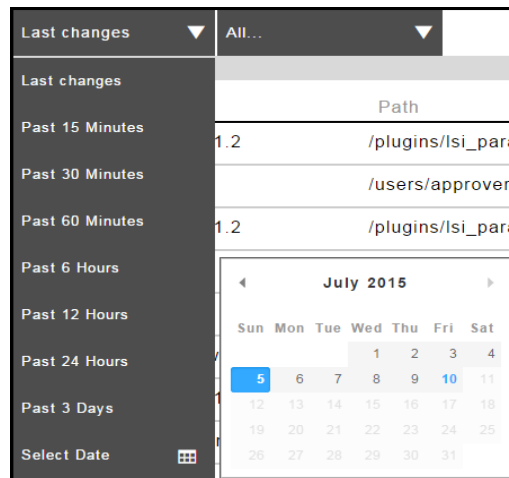
2. To select a time range for the search:

- a. Click the down arrow in the **Time range** field to open the drop-down list.
- b. Select a time range.
- c. If you want to use a time increment longer than three days, do the following:

- a. Click **Select Date**.

The Date Picker opens.

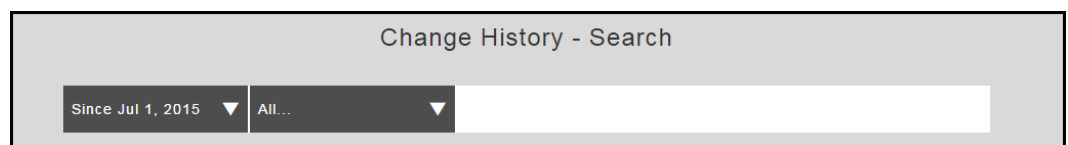
Example:



- b. Select a date.

The Date Picker closes and the date that you selected appears in the Time Increment field.

Example:



3. To select an objects for the search:

- a. Click the down arrow in the **Object** field to open the drop-down list.
- b. Select the objects for the search.

4. Enter the search criteria.

As you type, the system starts to search for objects that match your search criteria.

A list of objects matching your search criteria appears in the results section.

5. Select an object in the list.

Example:

Since Jul 1, 2015	All...	property	X
Access Control En...	ecscm-property-2.0.1.6...	/plugins/ecscm-property-2.0.1.65837	▲
	propertyviewer-1.0.1	/plugins/propertyviewer-1.0.1	
Access Control List	propertyviewer-1.0.1	/plugins/propertyviewer-1.0.1	
	ecscm-property-2.0.1.6...	/plugins/ecscm-property-2.0.1.65837	
Plugin	statetemplate_fullprope...	/server/ec_notifiertemplates/statetemplate_fullpropertypaths	
	propertyviewer-1.0.1	/plugins/propertyviewer-1.0.1	
Property	ecscm-property-2.0.1.6...	/plugins/ecscm-property-2.0.1.65837	
	artifactversionlocationp...	/projects/default/applications/heat clinic (killer app)/component...	
	artifactversionlocationp...	/projects/default/applications/jpetstore-aws-beanstalk/compone...	
	label	/server/ec_notifiertemplates/statetemplate_fullpropertypaths/la...	
	artifactversionlocationp...	/projects/default/applications/heat clinic store 1.1/components/...	
	artifactversionlocationp...	/projects/default/applications/heat clinic (killer app)/component...	
	artifactversionlocationp...	/projects/default/applications/heat clinic store 1.1/components/...	
	artifactversionlocationp...	/projects/default/applications/heat clinic store 1.1/components/...	
	series 1 property function	/projects/electric cloud/schedules/report recent job trend/series...	
	artifactversionlocationp...	/projects/default/applications/heat clinic (killer app)/component...	

The change history for the object that you selected appears.

Example:

Change History for propertyviewer-1.0.1						
<div> <div>▼ Last changes</div> <div> <div>7/10/15 - 1:02 PM</div> <div>Now - 7/10/15 - 6:18 PM</div> <div>Most Recent</div> </div> </div>						
View All Changes	When ▲ ▼	What	Name	By...	Change	Path
Objects						
✓ Acl (1)	✓ 1 Jul 10, 2015 1:02 PM Pacific...	acl	propertyviewer-1.0.1	admin	created	...
✓ Ace (3)	✓ 2 Jul 10, 2015 1:02 PM Pacific...	ace	propertyviewer-1.0.1	admin	created	...
Changes						
✓ Created (4)	✓ 3 Jul 10, 2015 1:02 PM Pacific...	ace	propertyviewer-1.0.1	admin	created	...
Changed by...						
✓ Admin (4)	✓ 4 Jul 10, 2015 1:02 PM Pacific...	ace	propertyviewer-1.0.1	admin	created	...

Reverting Changes to Objects

You can revert changes that were made to an object and to an objects and its children.

Follow these steps to select the changes that you want to revert:

1. Go to the Change History.
2. Configure the filters to view specific changes in the Change History.

If View All Changes is selected, click it to remove the check mark next to it.

Select only the objects, change types, and the users or groups who made the changes. A check mark appears next to the filter criteria that you select.

While selecting changes, make sure to be aware of the number of changes.

3. Choose an object in the Change History.
4. Click the **View** button to view the change details.

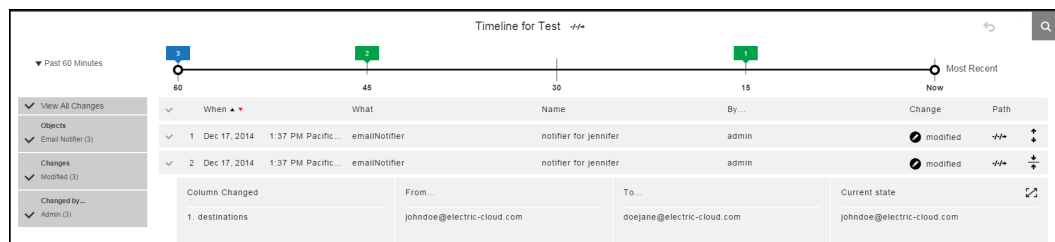
Example:



This information appears:

- Name of the child object (**Column Changed**)
- State of the child object before the change (**From**)
- State of the child object after the change (**To**)
- Current state of the child object (**Current state**).

Example:



5. Choose a change to revert.

6. In the **To** column, select the row of the object that you want to revert.

Example:

2 Dec 17, 2014 1:37 PM Pacific... emailNotifier notifier for jennifer admin modified			
Column Changed	From...	To...	Current state
1 destinations	john.doe@electric-cloud.com	doejane@electric-cloud.com	john.doe@electric-cloud.com

The Revert button is now available (enabled).

Example:



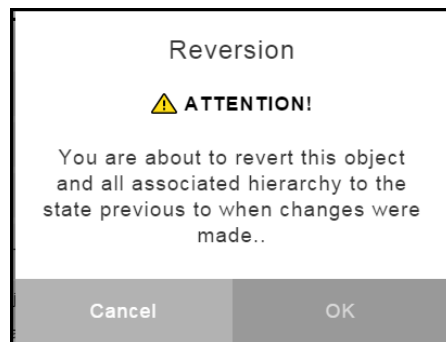
Repeat this step for each change that you want to revert or export.

7. To revert the changes, do the following steps:

- a. Click the Revert button.

A message confirming that you want to revert the selected object appears.

Example:



- b. If you want to revert the selected changes, click **OK**.

If the changes are successfully reverted, a message that the current state of the selected objects are successfully reverted appears.

If the changes are not successfully reverted, a message that the current state of the selected objects are not successfully reverted appears.

- c. If you do not want to revert the selected changes, click **Cancel**.

Change History Search Form

How to get here: From most pages, click the **Search** button to open the "Change History - Search" form.

Example:

The "Change History - Search" form has the following information:

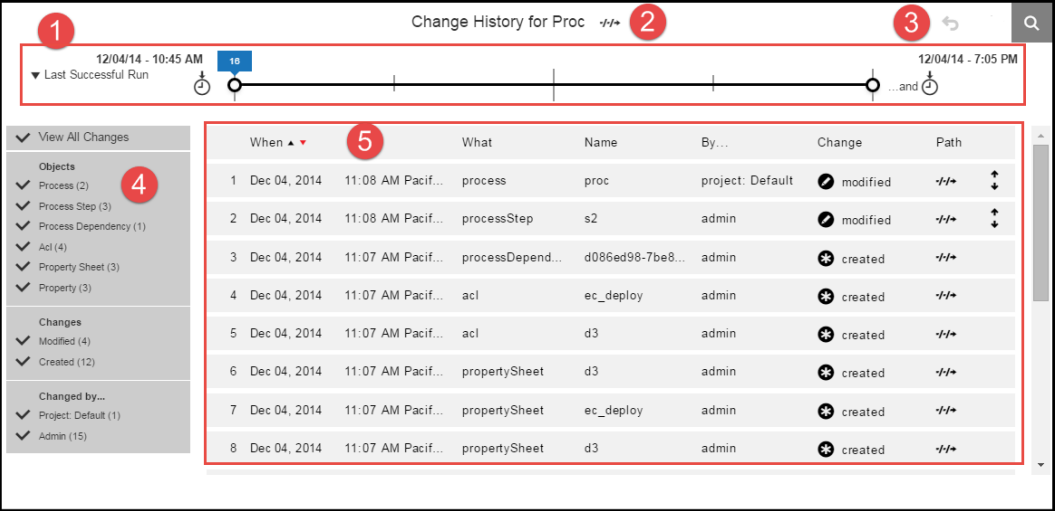
1	<p>Time range field.</p> <p>Click the down arrow to open the drop-down list of start times.</p> <p>The end time is the current time.</p>
2	<p>Objects field.</p> <p>Click the down arrow to open the drop-down list of objects to include in the search. You can select All or specific objects.</p> <p>By default, seven of the most commonly tracked objects are selected.</p>
3	<p>Search criteria.</p> <p>After you type, the system starts searching for objects based on the time range and objects that you selected.</p> <p>The search results are in the Change History.</p>

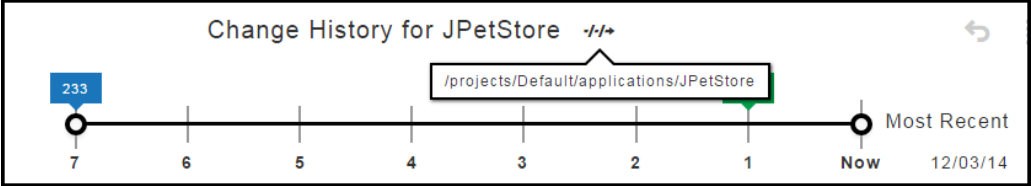
Change History Page

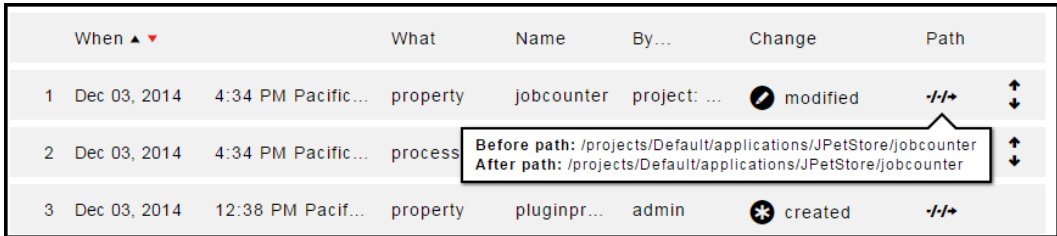
How to get here: Click the **Change History** button for a tracked object.

Example:

The Change History has this information about the object called *Proc*:

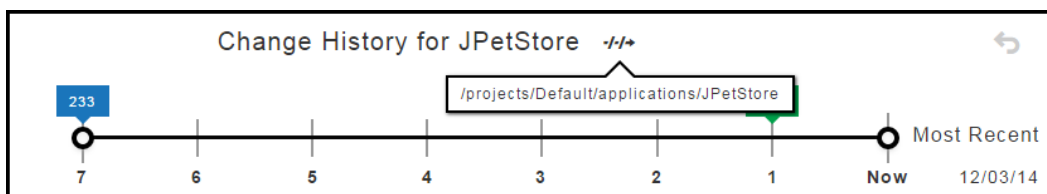


1	<p>Time line</p> <p>You can modify the start and end times.</p> <p>Default:</p> <ul style="list-style-type: none">• The entire time line is selected. All changes appear in the Change History list.• The time increment is from the Last Successful Run to the most current change.• The start time is based when the last successful run occurred.• The end time is when the most recent changed occurred.
2	<p>Path to the tracked object.</p> <p>Example:</p> <p>Change History for JPetStore</p> <p>233</p> <p>/projects/Default/applications/JPetStore</p> <p>7 6 5 4 3 2 1 Now</p> <p>Most Recent 12/03/14</p>
3	 <p>Click to revert the selected changes.</p>

6	<p>Time line.</p> <p>The start time is based on the time range that you selected.</p> <p>The end time is the current time.</p> <p>You can manually change the start and end times after you run the search and get the search results.</p>
4	<p>Filters for the change history.</p> <p>You can view all changes or view only selected changes.</p> <p>The objects in the list are the objects in the change history search results.</p>
5	<p>Change history for the selected object.</p> <ul style="list-style-type: none"> • When—the date and time that the object changed. • What—The type of object. • Name—The name of the object. • By—The "user" that changed the object, which can be a project or a user. • Change—The type of change. • Path—Click the View Path button to see the path to the object.  <p>Click the View button to view more information about a specific change in the change history.</p>

Paths to Objects

Click the **View Path** button next to the "Change History for JPetStore" title to see the path to the application.



Click the **View Path** button to see the change in the path to the object before and after the change.

	When ▲ ▼		What	Name	By...	Change	Path	
1	Dec 03, 2014 4:34 PM Pacific...	property	jobcounter	project: ...	modified	-/+	↑ ↓	
2	Dec 03, 2014 4:34 PM Pacific...	process	Before path: /projects/Default/applications/JPetStore/jobcounter After path: /projects/Default/applications/JPetStore/jobcounter					↑ ↓
3	Dec 03, 2014 12:38 PM Pacif...	property	pluginpr...	admin	created	-/+		

Detailed Object Changes

Click the **View** button to see the change in the property called emailNotifier.

Timeline for Test -/+

▼ Past 60 Minutes

3

2

1

Now

Most Recent

View All Changes

Objects

Email Notifier (3)

Changes

Modified (3)

Changed by...

Admin (3)

▼ When ▲

What

Name

By ...

Change

Path

1

Dec 17, 2014

1:37 PM Pacific...

emailNotifier

notifier for jennifer

admin

modified

▼/+

↑

↓

2

Dec 17, 2014

1:37 PM Pacific...

emailNotifier

notifier for jennifer

admin

modified

▼/+

↑

↓

Column Changed

From ...

To ...

Current state

↗

1. destinations

johndoe@electric-cloud.com

doejane@electric-cloud.com

johndoe@electric-cloud.com

Click the **Expand** button to all the changes to the property,

Example:



When you click the **Expand** button in a cell, you can see more details in the current change in the cell.

If you click the **Select All** button, all the changes about the object appear.

✕	← 2	Dec 17, 2014 1:37 PM Pacific Standard Time	emailNotifier	notifier for jennifer	By: admin	modified	-/-+ ↕
Column Changed	From...	To...	Current state				↗
1. destinations	doejane@electric-cloud.com	doejane@electric-cloud.com	johndoe@electric-cloud.com				

To select an object to revert or import the changes to an XML file, select the row of the object in the expanded view.

✓	2	Dec 17, 2014 1:37 PM Pacific...	emailNotifier	notifier for jennifer	admin	modified	-/-+ ↕
Column Changed	From...	To...	Current state				↗
1. destinations	johndoe@electric-cloud.com	doejane@electric-cloud.com	johndoe@electric-cloud.com				

Credentials and User Impersonation

[What is a Credential?](#) on page 973

[Defining a Credential](#) on page 973

[Why Use Credentials?](#) on page 973

[Credential Access Control](#) on page 974

[Using Credentials for Impersonation](#) on page 974

[Accessing Credentials From a Step](#) on page 976

[Credential References](#) on page 977

[Best Practices for Retrieving Credentials](#) on page 978

What is a Credential?

In ElectricFlow, a "credential" is an object that stores a user name and password for later use.

Two credential types are available, *stored* or *dynamic*:

- **stored** credentials - These credentials are given a name and stored in encrypted form in the database. Each Project has a list of stored credentials it owns. These credentials are managed from the Project Details page.
- **dynamic** credentials - These credentials are captured when a job is created. Dynamic credentials are stored on the server temporarily until the job completes and then discarded.

Defining a Credential

After a credential is created, no one can view the password for the credential's account. This means one person can define a credential and enter the password, and other people can use the credential (and its account) without needing to know the password.

To create a new credential in the automation platform UI:

1. Click the Projects tab.
2. Select a project (first column) to see the Project Details page.
3. Select the Credentials tab and then click **Create Credential**.

Click the **Help** link on the New Credential page if you need more information about what to enter in the fields.

Why Use Credentials?

Use ElectricFlow credentials for:

- User impersonation
- Saving passwords for use inside steps

When a step needs to run as a particular user, ElectricFlow can retrieve the username and password from a stored credential. The credential is passed to the agent over an encrypted channel so the agent can authenticate itself to the operating system and set up a security context where the step runs with the user permissions in the credential.

In some cases, a step needs to enter credentials directly to an application.

- The credential may be a fixed application credential that needs to be presented every time the step runs. In this case, a stored credential provides a place to put this information without needing to embed the password in a script.
- Other times, the password is not known in advance. In this case, a dynamic credential created when a job is launched can prompt a user for information and then pass the credential to the step in a secure manner.

Credential Access Control

Because ElectricFlow stores passwords in a recoverable manner, it is important that credentials are accessible only under carefully controlled circumstances. To protect credentials, ElectricFlow uses two mechanisms: access control lists (ACL) and *attached* credentials.

To use a credential for any purpose, a user who attempts to reference the credential must have execute permission on the credential object—this allows you to explicitly control which users are allowed to use a credential anywhere in the system.

In addition to the user-based access control check, ElectricFlow also requires a credential be explicitly *attached* to the object that is going to use it. After an object has a credential attached, object modifications are restricted to users who have both execute permission on the credential and modify permission on that object. This safeguard helps prevent credentials from accidentally being used by unauthorized users.

Attach a credential to an object in the automation platform UI one of these ways:

- Set the impersonation credential for a step, procedure, project, or schedule.

Impersonation credentials are inherited, so attaching a credential to a container object like a procedure or a project implicitly attaches it to every object inside the container, such as all steps in a procedure.
- Explicitly attach the credential to a step or schedule using the UI or the `attachCredential` API.

To access credentials from a step, the credential must be attached directly to the step, or the credential must be passed as a parameter to the containing procedure and have the parameter attached instead.

Using Credentials for Impersonation

This information addresses the question, "When a job step is running on a resource, under which account is it running and how can I control that?" By default, a job step runs under the account used by the ElectricFlow agent that runs the step, which was chosen when the agent was installed. This approach works well in environments where it makes sense to run all jobs under a single user such as a "build" user. You install all agents to run as the desired user, and you do not have to worry about anything else—every step of every job runs as that user.

However, in other environments you may prefer to run different jobs, or even different job steps, under different accounts.

For example:

- If independent groups are sharing an ElectricFlow system, you may want each group to use a different account for its jobs.
- Or, you may need to run jobs as particular users to access ClearCase views for those users.

- Or, there may be certain steps that require special privileges.

For example, as part of your builds you may need to run a step that generates a certificate using privileged corporate information; that step must run under a special high-privileged account, but you want the remainder of the steps in the build to use a less-privileged account.

ElectricFlow allows you to select accounts on a per-job or per-job-step basis. This mechanism is called *impersonation*, where the ElectricFlow agent impersonates a particular user for the duration of a job step, and this *impersonation* is implemented using credentials. The use of credentials creates special security challenges.

Note: It is important to ensure privileged accounts can be used only for the purposes you intend and cannot be "hijacked" for other purposes. The ElectricFlow access control mechanism contains special facilities to ensure proper credential use.

Setting the Impersonation Credential in the Automation Platform

Attaching Impersonation Credentials to Steps, Procedures, and Projects

You can attach impersonation credentials to procedure steps, procedures, and projects before executing a job step.

ElectricFlow searches for a credential in the following order and uses the first credential it finds:

1. Procedure step
2. Procedure to which the step belongs
3. Project to which the project belongs

If the step is in a nested procedure, and no credential is found on the step, its procedure, or its project, ElectricFlow checks the calling step (and its procedure and project), its caller, and so on until it has worked up through the topmost procedure in the job. If no credential is found, the job step runs under the ElectricFlow agent account on its resource.

This approach makes it easy to manage your account usage. For example, if you want all jobs in one project to use a particular account, define a credential in that project for the account, and then attach the credential to the project. If you want all jobs in a project to use a particular account, except for one step that should use a different account, create two credentials in the project. Attach the first credential to the project and attach the second credential to the particular step.

Attaching Impersonation Credentials to Schedules

You can also attach a credential to a schedule. The credential in the schedule is used for steps where no other credential is available.

The schedule's credential receives the lowest priority.

Attaching Impersonation Credentials to Jobs

You can specify a credential when you launch a job manually one of these ways:

- Provide the name of an existing credential
- Enter an account and password.

If you choose the second approach, the account and password are stored only in ElectricFlow for the duration of the job.

For either way, the credential you specified when you launched the job is used as a last resort for steps only with no other credential.

Attaching a New Impersonation Credential in the Automation Platform

To attach a new impersonation credential for a procedure, step, or schedule,

1. Click the Projects tab.
2. Select a project.
3. Go to the Project Details page for the project you selected.
4. Do one of the following:
 - a. For a **procedure**:
 - i. Click **Create Procedure** to go to the New Procedure page.
 - ii. On this page, select the Impersonation Credential drop-down box to select a credential to use for impersonation.
 - b. For a **step**:
 - i. Select a procedure to go to the Procedure Details page.
 - ii. Select a step name to go to the Edit Step page.
 - iii. On the Edit Step page, scroll down to the Impersonation section.
 - c. For a **schedule**:
 - i. Select the Schedules subtab.
 - ii. Select a schedule to go to the Edit Schedule page.
 - iii. Scroll down to Impersonation Credential section.

For more information in the automation platform UI, each of these web pages contains a **Help** link in the upper-right corner.

Accessing Credentials From a Step

The ElectricFlow server supports an operation called `getFullCredential` that can return a password when called from inside a step. The ectool interface for this operation is:

```
getFullCredential <credential name>
[--value <password|user>]
```

Without the `--value` option, the response is the same as that returned by `getCredential`, with the addition of a "password" element.

With `--value`, the simple text password or username value is returned on *stdout* so it can be used directly without XML parsing.

The `getFullCredential` API is allowed only inside a running step, and is allowed to use credentials that were explicitly attached to the step only.

Attaching a Credential to a Procedure Step, Procedure, or Schedule

A credential must be explicitly attached to the object using it so the server can perform an access control (ACL) check at definition time and limit the visibility of the password. To support accessing credentials other than the one being used for impersonation, steps, procedures, and schedules contain a list of credential names.

In the automation platform UI, to attach a credential (other than an impersonation credential) to a step, procedure, or schedule:

1. Go to an "Edit" page.

For example, to create a credential for a step, a step must already exist.

2. Select a step that needs a credential. The Edit Step page opens.
3. Scroll to the lower section of the page and click **Attach Credential**.

If you are an ectool command-line user, use the `attachCredential` API command. For more information on the command-line interface, go to the *ectool* Help topics within the online Help system.

Passing Credentials as Parameters

Sometimes you may not know which credential you need to use when you define a procedure step. In this situation, you can leave the credential choice up to the caller of the procedure.

The following example describes the procedure for passing credentials as parameters:

1. Create a project named InnerProj.
2. Within that project create a procedure named InnerProc with a parameter 'cred' of the type credential.
3. Within that procedure create a step named InnerStep with:
 - Command: `ectool getFullCredential cred`
 - Attached Parameter Credential: cred
4. Create a project named OuterProj with two credentials: cred1 and cred2.
5. Within that project create a procedure named OuterProc with two parameters, cred1 and cred2, of the type credential.
6. Within that procedure create two subprocedure steps that both call InnerStep from the InnerProj project:
 - OuterStep1 with Attached Parameter Credential cred1, and cred1 in the Parameters section
 - OuterStep2 with Attached Parameter Credential cred2, and cred2 in the Parameters section

If you are an ectool command-line user, you can use the `createFormalParameter` and `attachParameter` API commands. For more information on the command-line interface, go to the *ectool* Help topics within the online Help system.

Credential References

For schedules, the value for any actual parameter passed to a credential-type formal parameter is interpreted as a reference to a previously attached credential on the schedule.

For procedure steps, the value for any actual parameter passed to a credential type formal parameter is interpreted as a reference to a previously attached credential or credential parameter on the calling step.

When a job step is created, all credential references are resolved and the content is copied into transient credentials on the job step. Credential parameter references are resolved by looking for a credential in the calling job step's transient credential map—looking for a credential with the name specified as the actual parameter value.

Credential references in the API can be specified as a relative or an absolute credential path.

- A *relative* path is similar to "rootCred" and is interpreted as referring to a credential in the current project.
- An *absolute* path includes a project name like `"/projects/MyProject/credentials/rootCred"` and can refer to a credential in any project.

Anywhere a credential reference is accepted (for example, setting an impersonation credential or attaching a credential to a step), either form can be used.

Best Practices for Retrieving Credentials

The ElectricFlow impersonation mechanism provides powerful mechanisms for using different accounts in job steps, and it can be used to handle a variety of challenges. However, if misused, impersonation can open up dangerous security loopholes. This section discusses these risks and how to avoid them.

Use of credentials and impersonation tend to fall into two classes.

First Class

In the first class, your goal is to open up access to accounts, you want to make some accounts generally available, and you trust large groups of users (such as everyone who can access a particular project) to use the accounts appropriately. This usage type is simplest and relatively safe.

1. Define an ElectricFlow group containing all trusted users.
2. Set permissions on the Project so only trusted users can access the project.
3. Define credentials for accounts in the Project and grant execute permission to everyone in the group.

After completing this task, everyone in the group can use the accounts for arbitrary purposes. No one outside the group will be able to access the project or the credentials.

Second Class

In the second class, you want to provide tightly-controlled access to a highly privileged function such as generating a certificate or accessing sensitive information. This kind of usage requires careful thought to avoid allowing unintended access to the credential. The best way to implement a solution for this scenario is to restrict access to the credential and use it on individual steps only after careful analysis of issues such as the following:

Does the step's command use properties?

If so, someone could potentially change the property value to change the step behavior.

For example, suppose the step's command is defined like this:

```
echo ${/myProject/x}
```

Anyone with write access to property `x` can hijack the step by placing the value `"nothing; myCommand"` in the property.

After property substitution, the final command will be:


```
echo nothing; myCommand
```

which will cause "myCommand" to execute in the step.

Either restrict access to the property sheet or avoid substitutions.

Does the step execute code or commands that can be modified?

In most jobs that execute build and test sequences, the job extracts various files from a source code control system and some of those files contain code executed during the build. In this case, anyone with access to the source code control system can affect the executed code.

For example, suppose a step runs a Make or Ant command using a configuration file extracted from the source code control system. Anyone with access to the source code control system can modify the configuration file to introduce new commands executed during the Make or Ant step. In this situation, it is virtually impossible to control what happens during the step, so you should never attach a credential for a privileged account to such a step.

The higher you attach a credential, the greater the risk of uncontrolled access to the credential's account.

For example, if you attach a credential to a project, there is a good chance anyone with write access to the project, or even the ability to execute the project's procedures, can hijack the credential using one of the loopholes described above. If you attach a credential to a step that invokes a subprocedure, you effectively give anyone with write access to the subprocedure complete access to the credential's account. If you care about access to an account, you should use its credential in the narrowest possible fashion, such as attaching it to a single job step.

Avoid passing passwords on the command line

When using credentials, avoid passing the password on the command line because [on most platforms] those passwords will be visible to all users logged into the system. The secure way to pass a credential to an application is to use a secured file in the file system or to pipe the value into the application via `stdin`.

Defect Tracking

The Defect Tracking plugin enables linking existing defects to an ElectricFlow job. "Existing defects" are those defects previously created in the defect tracking system and already associated with the ElectricFlow job in some way.

For example, a defect is associated with a ElectricFlow job if the fix for the defect is part of the source code snapshot being built and tested in the job.

ElectricFlow pre-installs numerous defect tracking plugins including Bugzilla, ClearQuest, Fortress, JIRA, Quality Center, Ration Team Concert, Rally, Team Foundation Server, TeamForge, TestTrack, and more, using plugin integrations.

The following examples and instructions are for a JIRA integration. The steps for other defect tracking systems are similar to those for JIRA.

Scenario example

A developer fixes a defect in the "ABC" project and checks in the fix to the source control system, along with a comment that includes the fixed defect ID, for example:

```
ABC-123: fixed EFG bug
```

When the next ElectricFlow job is triggered for the ABC project, ElectricFlow checks out a source code snapshot from the source control system and queries the source control system for a log containing check-in details. This log, which should contain the above comment, will be stored in a property on the job.

The JIRA plugin will then do the following:

- Parse the property containing the source control system log to identify defect IDs.
- Query the configured JIRA server with identified defect IDs, including "ABC-123".
- Construct a descriptive URL to point to the JIRA defect on the JIRA server.
- Include the URL in the JIRA Report.
- Link to this JIRA Report from the Job Details page.

Example: Enabling the JIRA integration in your procedure

To ensure ElectricFlow links existing defects to a job, create a step to link the defects.

Go to Projects > select a Project > select a Procedure. To create a New Step, select the **Plugin** link.

- In the Choose Step panel, select Defect Tracking from the left pane, then select the defect tracking system you configured.
- The right-pane now shows the types of steps available for your configuration. Select the step you need and automatically go to the New Step page.
- On the New Step page, notice the Subprocedure section now contains the defect tracking integration you configured and the step you chose.

On the New Step page, enter information in the following fields:

General section:

Name—Enter a unique name for your subprocedure step (any name of your choice).

Description - (optional) A plain text or HTML description for this object. If using HTML, you must surround your text with `<html> ... </html>` tags. The only HTML tags allowed in the text are: `<a>` `` `
` `<div>` `<dl>` `` `<i>` `` `` `<p>` `<pre>` `` `<style>` `<table>` `<tc>` `<td>` `<th>` `<tr>` ``

Subprocedure section:

Resource—(optional) You do not need to change the resource you previously chose to run the procedure.

Parameters section:

config—(required) This is the name of the Defect Tracking configuration you created on the New Defect Tracking Configuration page.

Note: If you did not previously create a Defect Tracking configuration, you must do that now before you proceed. Go to Administration > Defect Tracking > click the **Create Configuration** link to see the New Defect Tracking Configuration page. Access the **Help** link on that page for more information.

prefix—(optional) This is the key used by JIRA as the prefix for defects within a project. If the "prefix" field is blank, a regular expression is used to try and match defect IDs.

propertyToParse - (optional) This is the property or property sheet to search for defect IDs. If the "propertyToParse" field is blank, the default property `/myJob/ecscm_changeLogs` is used.

Advanced section:

You do not need to enter any information in these fields.

The following screen is an example of the New Step page where you will create your Defect Tracking subprocedure step.

The screenshot shows the 'New Step' configuration page with the following sections:

- General:**
 - Name: [Text Field]
 - Description: [Text Area]
- Command:**
 - Subprocedure: EC-DefectTracking-JIRA_LinkDefects Change
 - Resource: [Text Field]
- Parameters:**
 - Configuration Name: [Text Field] (Required)
 - Prefix: [Text Field]
 - Property To Parse: [Text Field]
- Advanced:**
 - Precondition: [Text Area]
 - Run Condition: [Text Area]

ElectricSentry

ElectricSentry is the ElectricFlow **engine** for continuous integration, integrating with Source Control Management (SCM) systems. ElectricSentry is installed automatically with ElectricFlow and is contained in an ElectricFlow plugin named ECSCM. When ElectricSentry detects a new or modified source file check-in, it launches a user-defined procedure to build and test the latest version of source files. Using a few simple properties, the entire process is controlled by the user who determines when ElectricSentry is active and which projects and branches are monitored and built.

While you can interact with ElectricSentry directly, you may want to use the Continuous Integration Dashboard, *the front-end user interface* for ElectricSentry, which provides a visual display of your builds, and automates the configuration process for your branch, project, or procedure so you can start using continuous integration schedules quickly.

This Help topic describes using ElectricSentry directly, and NOT the Continuous Integration Dashboard, which has its own Help topic for dashboard use. To get started with the "dashboard", select the Home > Continuous Integration tabs.

How ElectricSentry works

ElectricSentry finds procedures the user wants to monitor by scanning for a special type of ElectricFlow schedule. From the schedule, ElectricSentry retrieves the name of the procedure to run, along with values to use as actual parameters when the procedure runs.

ElectricSentry can run at a regular interval you determine. When it runs, ElectricSentry scans all ElectricFlow projects, looking for schedules configured for use with ElectricSentry. When it finds one, ElectricSentry checks to see if there is already a job running based on that schedule and will not start a second job for the same schedule until the previous job completes.

Next, ElectricSentry queries the SCM system to see if new check-ins have occurred since the last attempted build. If the most recent check-in is different from the last attempted build, ElectricSentry then checks a "quiet period" by comparing the current time to the check-in time. If the user-configured time interval has elapsed since the most recent check-in, ElectricSentry runs the procedure to start a new job.

Finally, ElectricSentry deletes previous runs of itself, so only the most recent run appears in the jobs list. If using the Continuous Integration Dashboard, this is slightly different—you will see the previous 10 builds that ran.

Configuring ElectricSentry

ElectricSentry does not require any specific configuration, but it does allow fine tuning for some key settings. To change any setting, you must be logged into ElectricFlow and then navigate to the Electric Cloud project.

Quiet time

When setting up a continuous integration system, it is common to require an inactivity period before starting a build. This time period allows developers to make multiple, coordinated check-ins to ensure a build does not start with only some of the changes—assuming all changes are checked-in within the specified inactivity time period. This time period also gives developers an opportunity to "back-out" a change if they realize it is not correct.

With ElectricSentry, the inactivity time period can be configured globally for all projects or individually for a single project. The global setup is stored in the ECSCM-SentryMonitor schedule in the Electric Cloud project, using a property named ElectricSentrySettings/QuietTimeMinutes. You can set this property to the number of minutes you prefer for the quiet time prior to starting a build. If you remove this property, ElectricSentry will use a five-minute default time.

Resource

The ECSCM plugin contains a procedure named `ElectricSentry`. This procedure has a parameter named `sentryResource` that determines where `ElectricSentry` will run. The parameter defaults to a resource named *local*, which is created by the `ElectricFlow` installer. The local resource is configured to run on the machine named *localhost*, which refers to the machine where the `ElectricFlow` server is running.

You may need to change the resource where `ElectricSentry` runs, if for example, the server machine is not configured to run your SCM tools. You can change this configuration by modifying the `ECSCM-SentryMonitor` schedule to specify a different resource in the `sentryResource` parameter.

To do this:

1. Go to the Projects page.
2. Select the Electric Cloud project name to go to the Project Details page.
3. Click the Schedules subtab.
4. Select the `ECSCM-SentryMonitor` schedule.
5. Change the `sentryResource` field to the name of the resource (or the resource pool name) where you want `ElectricSentry` to run.
6. Click **OK** to save your new information.

Polling frequency

`ElectricSentry` is set to look for new check-ins every five minutes. If you want `ElectricSentry` to check more or less often, open the `ECSCM-SentryMonitor` schedule in the Electric Cloud project. Change the number of minutes to speed up or slow down the `ElectricSentry` polling frequency entry.

Time-of-day and day-of-week

The initial setting allows `ElectricSentry` to run between 7 am and 11 pm every day of the week, but you can change this time period to monitor check-ins at different hours. The time period should cover the most likely hours developers would be checking-in changes, then frees the resources (after the last job started by `ElectricSentry` finishes) to run overnight builds. You can modify the `ECSCM-SentryMonitor` schedule to change the hours or days `ElectricSentry` monitors check-ins.

Configuring a build for continuous integration

You can control which projects and branches `ElectricSentry` monitors for continuous integration by creating a schedule that runs a procedure and by adding `ElectricSentry` settings. These tasks are made easier using the Continuous Integration Dashboard.

Source Control Management (SCM) configurations

`ElectricFlow` bundles and supports a number of source control types. After creating a source control configuration, your entry will appear in the table on the Source Control Configurations web page—to see this web page, select the Administration > Source Control tabs.

Click the **Create Configuration** link to configure your source control system.

Note: Electric Cloud has tested numerous SCM versions and where possible, all SCM integrations were created in a generic manner to avoid SCM release-specific differences. In most cases, Electric Cloud supports any version of Perforce, ClearCase, Subversion, AccuRev, CM Synergy, Borland's StarTeam, and many others.

Select the Administration > Plugins tabs to go to the Plugin Manager page to the current list of available SCM plugins. If you do not see the SCM you need, it may be available in the Plugin Catalog—select the View Catalog tab.

Create a procedure

Navigate to your project and create a procedure that checks out sources from your SCM system and then runs your build process. Electric Cloud recommends having the procedure take a "branch name" parameter, then use this single procedure to build multiple branches of the same project.

Create a schedule

In your project, create a schedule that calls the procedure and specifies parameters to call the procedure. On the Project Details web page, select the Schedules tab and click the **CI Configuration** link to go to the New CI Configuration page. Enter a name for the new schedule and choose the SCM Configuration you want to use. For more information on this page, click the **Help** link in the upper-right corner of the screen. If you set up your procedure with a parameter for a branch, create one schedule for each branch you want to monitor.

Optional - running ElectricSentry on multiple resources

ElectricSentry normally runs as a single procedure executing queries against all SCM systems configured by ElectricSentry "trigger schedules." Because all queries are executed in a single step, these queries are executed by the same ElectricFlow resource. This process works very well for many companies, but for some large enterprise companies this process is not adequate.

You can benefit from using ElectricSentry on multiple resources if any of the following items apply to your organization:

- Connectivity - No one single resource has access to all SCM systems in your organization.
- Security/Authentication - Specific resources may contain Authentication tickets required for access to certain parts of your SCM depots, and no single resource contains all of these tickets.
- Scalability - You may need too many trigger schedules to monitor in a single execution interval, which can cause unnecessarily long execution times for ElectricSentry jobs.
- Scheduling - ElectricSentry is most useful during the work day when developers are adding new code check-ins. At night, it is common to redeploy resources to do nightly builds. If ElectricSentry runs on a central resource that serves development teams in multiple time zones, there is no way to schedule independent Sentry operation for different time zones.

Overview

The solution to all of these issues [above] is to run multiple ElectricSentry instances concurrently.

- Each Sentry instance can choose its own resource and its own operation schedule.
- Each instance can be restricted to monitoring schedules for one or more ElectricFlow projects.
- Also, you can create a global instance to monitor all projects not monitored by a restricted instance.

Each ElectricSentry instance is implemented by re-using the ElectricSentry procedure in the ECSCM plugin. This procedure has two parameters, `sentryResource` and `projectList`, that allow you to specify the configuration for different instances.

Each instance will have its own copy of the SentryMonitor schedule—one for each resource where you want ElectricSentry to run.

The default configuration has no projects specified in the "projectList" parameter—this is called the *global* instance. The global instance monitors all projects not explicitly monitored by any other instance. If a global instance is your only instance, it will monitor all projects.

Configuring ElectricSentry to use multiple resources

When ElectricFlow is installed, it creates a project called Electric Cloud that contains ElectricSentry's Sentry Monitor schedules. By default, all steps will run on the resource named local and all projects will be monitored. To use multiple resources, you need to create new schedules in the Electric Cloud project, creating new ElectricSentry instances.

1. Navigate to the Electric Cloud project and select it.
2. Select the Schedules tab.
3. Disable the schedule named ECSCM-SentryMonitor.
4. Click the **Copy** link next to the ECSCM-SentryMonitor schedule - **repeat** for each new ElectricSentry instance you want to create.
5. Re-enable the schedule named ECSCM-SentryMonitor.
6. Navigate to one of the new schedules you created and select it to edit it - change the name from ECSCM-SentryMonitor copy n to something more meaningful, for example, "ECSCM-SentryMonitor for ABC team".
7. In the Parameters section:
 - projectList box - enter the project names you need to monitor - one project name per line.
 - sentryResource box - enter the resource or pool name where you want this ElectricSentry instance to run.
8. In the Frequency section:
 - Adjust the "time of day" and "days of the week" settings to schedule this ElectricSentry instance to run.

Note: If you are setting up ElectricSentry to run in a different time zone, make sure you set the Time Zone field for the location where ElectricSentry will run.

- a. In the Advanced section, click the Enabled check box.
- b. Click the **Save** button.

Note: For each new schedule you create, repeat steps 6-10 above.

The Job Step Execution Environment

Terms and definitions

Various factors influence job step execution. Both a job step and its postprocessor (if any) run in the same environment, except shell. The shell used to run a command comes from what is specified in the step. If a shell is not specified on the step, the shell specified on the resource is used— this is also the same shell used for running the postprocessor.

Machine

The machine where a job step is executed is determined by the resource specified in the corresponding procedure step. If a pool is specified in the procedure step, ElectricFlow picks a specific resource from that pool. A job step can determine its actual resource by querying the property `/myResource/resourceName` or assigned `resourceName` property on the job step. The host where a step is expected is `/myResource/hostname`. If the resource is a proxy agent, this property contains the name of the proxy target. The agent host name is in `/myResource/proxyHostName`.

OS-level access control

For its resource, a job step executes under the same account as the ElectricFlow agent. You may want to contact Electric Cloud technical support for help configuring ElectricFlow agents. Basically, the agent needs to know what account it will run as, and Windows agents require additional setup if impersonation is used. If using impersonation, the job step runs under the credential effectively attached to the step.

Environment variables

For its resource, a job step inherits environment variables from the ElectricFlow agent. The agent's environment variables can be configured as part of the agent configuration. In particular, the `PATH` environment variable typically includes the ElectricFlow installation directory for easy access to applications such as *ectool* and *postp* (the ElectricFlow postprocessor).

Working directory

The default job step working directory is the top-level directory in the job's workspace. However, you can configure the working directory with a property on the procedure step. When you run on a proxy agent, the step actually runs on the proxy target, in the working directory specified in the step. If the working directory is not specified, the step runs in the UNIX path to the workspace.

Standard I/O

Standard job step output and errors are both redirected to the step log file.

ElectricFlow access

For easy access to ElectricFlow, the following environment variables are set automatically for a job step:

COMMANDER_HOME

A variable whose value is the base installation directory for an agent. On Windows, this directory is typically `C:\Program Files\Electric Cloud\ElectricCommander`. On UNIX platform, this directory is typically `/opt/electriccloud/electriccommander`.

COMMANDER_SERVER

IP address for the ElectricFlow server machine.

COMMANDER_PORT

Port number for normal communication with the ElectricFlow server.

COMMANDER_PLUGINS

(configurable) The directory where installed plugins live (for example, C:\Documents and Settings\All Users\Application Data\Electric Cloud\ElectricCommander\Plugins)

COMMANDER_HTTPS_PORT

Port number for secure communication with the ElectricFlow server.

COMMANDER_JOBID

Unique identifier for the job containing the current job step.

COMMANDER_JOBSTEPID

Unique identifier for this job step.

COMMANDER_SESSIONID

ElectricFlow session identifier for the current job, which allows ectool to access ElectricFlow with job associated privileges, without an additional login.

COMMANDER_WORKSPACE_UNIX

Absolute path to the workspace directory for this job, in a form suitable for use on UNIX machines.

COMMANDER_WORKSPACE_WINDRIVE

Absolute path to the workspace directory for this job, in a form suitable for use on Windows, and starting with a drive letter.

COMMANDER_WORKSPACE_WINUNC

Absolute path to the workspace directory for this job, in a form suitable for use on Windows, specified using UNC notation.

COMMANDER_WORKSPACE

Absolute path suitable for accessing the top-level workspace directory for this job on this machine. On a UNIX machine, this has the same value as **COMMANDER_WORKSPACE_UNIX**. For Windows, it is the same as **COMMANDER_WORKSPACE_WINUNC**.

COMMANDER_WORKSPACE_NAME

This is the name of the workspace on the ElectricFlow server.

These environment variables allow you to invoke ectool without specifying a `--server` argument. They also provide a context for accessing properties in ectool. For example, `"ectool getProperty foo"` looks for the property named "foo" on the current job step. Environment variables provide ectool with a session including all user privileges associated with the job.

Also, if the resource is a proxy agent, these environment variables are available in the step's environment on the proxy target.

Preflight Builds

[Preflight Build Solution](#)

[Preflights with ElectricFlow](#)

[Samples](#)

[Default SCMs](#)

[Troubleshooting](#)

Why use Preflight Builds?

When developers make changes, they generally build and test their code locally and *commit* their changes if the code ran successfully. Unfortunately, this process is not sufficient.

Code changes may break the production build because of environment differences, platform-specific issues, or incomplete commits. These issues leave the product in a broken state until the changes are backed-out of the product or the developer commits a fix, which often affects the entire development team. It is a time-consuming process to find the problem when multiple developers commit their changes simultaneously and the build breaks.

Preflight Build Solution

A **Preflight Build** is used to build and test a developer's changes before those changes are committed. A "post-commit" source tree is simulated by creating a clean source snapshot and overlaying the developer's changes on top of it. These sources are then passed through the production build procedure to validate the changes work successfully.

Developers are allowed to commit their changes only if the preflight build is successful. Because developer changes are built and tested in isolation, many common reasons for broken production builds are eliminated.

Preflights with ElectricFlow

Workflow

After a preflight build is configured, the general flow of interaction between the server, agent, and client is:

- A developer invokes the client preflight program either through an IDE or via the command-line. The following arguments are specified in the XML configuration file or on the command-line:
 - Define which ElectricFlow server to use (hostname, ports, and so on),
 - Define the project, procedure, and parameters to use to start the preflight build, and
 - SCM-specific information to use (ports, changelists, and so on).
- Next, the client preflight program connects to the ElectricFlow server and launches the specified procedure.
 - The job is started and the developer's changes are uploaded.
 - If auto-commit was turned on, the client program waits for the job to complete. Otherwise, the client program exits immediately.
- The job progresses until it gets to the special step added to the procedure to extract sources and overlay changes.
 - The agent-side driver creates a clean source snapshot based on configuration information passed to it from the client-side.
 - The step overlays changes [uploaded by the client] on top of the snapshot to simulate the developer's check-in.
- The build continues as usual, using the modified sources.
- If the developer chose to auto-commit changes, the preflight program was waiting for the job to complete. If the job is successful, the changes are committed if:
 - No files were added or removed from the change sets.
 - No files were modified since the preflight build started.
 - All conditions set in the driver script were met.

Components

Three components are required for an ElectricFlow preflight build:

- **Server** - This is the main ElectricFlow server that will run the preflight builds.
- **Agent** - This is any ElectricFlow agent machine that is setup as a resource for the server. This machine must be able to communicate with the source control system because it is responsible for creating a clean source snapshot on which the developer's changes are overlayed.
- **Client** - This is the machine where a developer has active code changes to submit for a preflight build. Like the agent, this machine must be able to communicate with the source control system because it is responsible for determining which files were modified so it can transmit these files to the agent.

Installation

Preflight is automatically installed with SCM plugins. ElectricFlow pre-installs numerous source code management (SCM) plugins including Accurev, Bazaar, CVS, ClearCase, Git, Mercurial, Perforce, SVN,

StarTeam, Team Foundation Server, and Vault. Before you can use your preferred SCM, you will need it to communicate with the ElectricFlow server.

Configuration

This section provides details for setting up and running Preflight builds in a typical build environment.

Server

Adding the preflight snapshot step

A typical production build procedure has a step that extracts a clean source code snapshot before the actual build starts. For preflight builds, a special step needs to run in place of this snapshot step. This special step is responsible for creating the base snapshot and then overlaying the developer's changes.

- On the Procedure Details page, to create a new step click the Plugin link.
- On the Choose Step panel, select Source Code Management.
- Choose the ECSCM plugin for your SCM or from the right-pane, select the "Preflight" step for your SCM.
- On the New Step page, notice that your SCM is displayed in the Subprocedure section.
 - Set the Resource name for the step to an agent that can communicate with the SCM system, so it can create the source snapshot.
- The Parameter section also displays appropriate fields to enter information for your SCM. The following two fields are common to most SCMs:
 - Configuration—Enter the name you created for your SCM configuration.
 - Destination Directory—This is a path relative to the job's workspace, where the source tree will be created.
- Enter all other information required for your new step.
- Click **OK**.

Choosing between snapshot steps

Only one snapshot step should run depending on whether a production or preflight build is invoked. To enable the appropriate step, add a checkbox-style parameter to the build procedure to determine whether or not the build is a preflight. Now, use that parameter in the Run Condition field on the two snapshot steps. In the following example, the parameter is named "preflight", with an unchecked value of "false" and a checked value of "true".

- For the preflight snapshot step, set the run condition to `${preflight}`
- For the production snapshot step, set the run condition to `[/javascript getProperty("preflight") == "false"]`

Choosing how files are uploaded

By default, when developers run preflight builds, their changes are uploaded to the job workspace via the server. However, if the job workspace is on a network share, accessible to the developer's machine, a useful optimization is for files to be copied directly into the job workspace. To use the network share approach, set a flag on the top-level procedure that is invoked when running a preflight build. This property can be created on the Procedure Details web page as follows:

1. Create a nested property sheet on the procedure called `ec_preflight`.
2. Create a property in that procedure called `waitForStep` with a value of "1".
This property can be created from the command-line by calling:

```
ectool setProperty ec_preflight/waitForStep 1 --projectName <project> --
procedureName <procedure>
```

Agent

The agent machine must be able to communicate with the SCM system to create the source snapshot.

Client

From a developer's machine, there are two ways to start preflight builds:

- From an IDE:
 - Eclipse
Open a Run dialogue, select Launch Commander Procedure, then select an existing preflight configuration or create a new one.
For more information, see the *ElectricCommander Eclipse Integration Tech Note* PDF file or the Eclipse plugin.
 - Visual Studio (See the *ElectricCommander Visual Studio Tech Note* PDF file or the Visual Studio plugin)
- From the command line:
The executable program, `ecclientpreflight`, is used to run a preflight build from the command-line. This program is included in the ElectricFlow "Tools" installation. When you run `ecclientpreflight`, configuration options can be passed in an XML configuration file or on the command-line.

If the same option is specified in both the configuration file and on the command-line, the value passed on the command-line takes precedence. The XML configuration file can be passed to the program via the `--config` option. If the XML configuration file is not passed in, current and ancestor directories are searched for a file named `".preflight"`, using the first one it finds as the configuration file.

Generally, a `".preflight"` file is stored at the root of a developer's workspace, containing configuration information specific to that workspace.

Samples

In the following samples, the SCM was set to Perforce.

The following table is output from "ecclientpreflight --help", enumerating the command-line options. Some options may be specified using an abbreviated format (`-c`, with a single dash) or a more verbose format (`--config`, with two dashes).

Electric Cloud Client-Side Preflight Tool, Version: 4.x Copyright (C) 2012 Electric Cloud, Inc. All rights reserved. Usage: ecclientpreflight [options]
General Options

<code>-c,--config <file></code>	Load configuration options from the specified XML file. If <code><file></code> is '-', read standard input. If not supplied, the current and ancestor directories will be searched for a .preflight file. The first .preflight found will be used as the preflight configuration.
<code>--load <file></code>	Evaluate <code><file></code> after option processing. May appear multiple times.
<code>-d,--debug</code>	Display debugging information.
<code>-v,--version</code>	Display version information.
<code>-h,--help</code>	Display this information.
Server Communication Options	
<code>--server <name></code>	The host name (defaults to localhost).
<code>--port <port></code>	The HTTP port (defaults to 8000).
<code>--securePort <port></code>	The HTTPS port (defaults to 8443).
<code>--secure</code>	If set, only secure connections are used. Off, by default.
<code>--timeout <timeout></code>	The response timeout in seconds (defaults to 180).
<code>--userName <name></code>	The name of the ElectricFlow user to login as.
<code>--password <password></code>	The password for the specified user. If blank and the user has no active session, a prompt to enter the password will be provided.
<code>--driverLocation</code>	Property path to a property sheet containing the driver scripts that will be loaded and run after a connection to the server is established. Defaults to <code>/server/ec_preflight</code> .
<code>--mainDriver</code>	The name of a property in the sheet specified by <code>driverLocation</code> . This is the main driver script that will be loaded and run after a connection to the server is established.
Logging Options	
<code>-l,--log</code>	If specified, debug information is logged. Off, by default.

<code>--logDir</code>	Where to store log and other files. Defaults to the user's home directory.
Procedure Invocation Options	
<code>--projectName</code> <code><name></code>	The name of the ElectricFlow project containing the procedure to invoke.
<code>--procedureName</code> <code><name></code>	The name of the ElectricFlow procedure to invoke.
<code>-p, --param</code> <code><name>=<value></code>	Enter additional parameters to the procedure. May appear multiple times. If any parameters were specified in the config file, those supplied on the command-line append to that list, overriding parameters with the same name.
<code>--priority</code>	The priority of the job. Possible values are low, normal, high, highest. If left unspecified, defaults to normal.
<code>--jobTimeout</code> <code><timeout></code>	The number of seconds to wait for the job to complete when auto-committing changes. Defaults to 3600 seconds (1 hour).
<code>--waitForJob</code>	Wait for the job to complete and report its outcome. This action does not occur by default unless a set of SCM charges are being committed automatically.
<code>--runOnly</code>	Run the procedure and exit immediately. The SCM driver is not downloaded in this case.
SCM Options	
<code>--scmType</code>	The name of the SCM, required, unless the procedure is invoked in 'run only' mode. The driver is downloaded from <code>\$driverLocation/clientDrivers/\$scmType</code> . Valid values: ECSCM-Accurev, ECSCM-Bazaar, ECSCM-ClearCase, ECSCM-CVS, ECSCM-Git, ECSCM-Mercurial, ECSCM-Perforce, ECSCM-Repo, ECSCM-StarTeam, ECSCM-SVN, ECSCM-TFS, ECSCM-Vault.
<code>--autoCommit <1 0></code>	Whether or not the changes should be automatically committed if the job completes successfully. Off, by default.
<code>--commitComment</code> <code><comment></code>	A comment for the auto-commit.

Perforce Options	
<code>--p4port <port></code>	The value of P4PORT. May also be set in the environment. This is a required value.
<code>--p4user <user></code>	The value of P4USER. May also be set in the environment. This is a required value.
<code>--p4passwd <password></code>	The value of P4PASSWD. May also be set in the environment.
<code>--p4client <client></code>	The value of P4CLIENT. May also be set in the environment. This is a required value.
<code>--p4template <template></code>	The name of a Perforce client used to create a base snapshot before overlaying local changes. Defaults to the value of <code>--p4client</code> if not specified.
<code>--p4changelist <change></code>	The changelist number (or default) whose changes are being tested. May be specified multiple times. If no changelists are specified, all client changelists will be used.

The following is a sample XML configuration file with all options specified:

```
<?xml version="1.0" encoding="utf-8"?>
<data>
  <server>
    <userName>myUser</userName>
    <password>myPass</password>
    <hostName>ElectricFlowServer</hostName>
    <port>1234</port>
    <securePort>2345</securePort>
    <stompPort>3456</stompPort>
    <secure>0</secure>
    <timeout>3456</timeout>
    <driverLocation>/server/ec_preflight/alternateDrivers</driverLocation>
    <mainDriver>myMainClientDriver</mainDriver>
  </server>
  <procedure>
    <projectName>myProject</projectName>
    <procedureName>myProcedure</procedureName>
  </procedure>
</data>
```



```

    <parameter>
      <name>branch</name>
      <value>main</value>
    </parameter>
    <parameter>
      <name>preflight</name>
      <value>true</value>
    </parameter>
    <priority>high</priority>
    <waitForJob>1</waitForJob>
    <jobTimeout>3600</jobTimeout>
  </procedure>
<scm>
  <type>perforce</type>
  <port>perf:1234</port>
  <user>myUser</user>
  <password>myPass</password>
  <client>myUser-main-client</client>
  <template>myUser-main-template</template>
  <changelist>default</changelist>
  <changelist>67382</changelist>
  <autoCommit>1</autoCommit>
  <commitComment>Fixing bug 38582.</commitComment>
</scm>
</data>

```

In most cases, a developer does not need to override all default values, and will probably want to specify passwords and some SCM-specific options on the command-line.

The following is a more typical XML configuration:

```

<?xml version="1.0" encoding="utf-8"?>
<data>
  <server>
    <userName>myUser</userName>
    <hostName>commanderServer</hostName>
  </server>
  <procedure>
    <projectName>myProject</projectName>

```

```
<procedureName>myProcedure</procedureName>
<parameter>
  <name>branch</name>
  <value>main</value>
</parameter>
<parameter>
  <name>preflight</name>
  <value>true</value>
</parameter>
<jobTimeout>3600</jobTimeout>
</procedure>
<scm>
  <type>perforce</type>
  <port>perf:1234</port>
  <user>myUser</user>
  <client>myUser-main-client</client>
</scm>
</data>
```

If this file is stored in a .preflight file at the top-level of a source tree, the command to start the preflight would look something like this:

```
eccllientpreflight --p4changelist 56793 --autoCommit 1 --commitComment "Fixing
bug 38582."
```

Default SCMs

ElectricFlow includes preflight support for some of the most common Source Control (SCM) systems. For an SCM plugin you installed, look for a Help link for that plugin (on the Plugins page). Command-line options are included in this Help topic for the following SCMs:

Acc uR ev	Git	Star Tea m
Baz aar	Me rcu rial	Sub vers ion

Cle arC ase	Pe rfo rce	TFS
CV S	Re po	Vau lt

Perforce

The command-line options for the Perforce driver for `ecclientpreflight` are:

Perforce Options / Descriptions	
<code>--p4port <port></code>	Required: The value of P4PORT may be set in the environment also.
<code>--p4user <user></code>	Required: The value of P4USER may be set in the environment also.
<code>--p4passwd <password></code>	The value of P4PASSWD may be set in the environment also.
<code>--p4client <client></code>	Required: The value of P4CLIENT may be set in the environment also.
<code>--p4template <template></code>	The name of a Perforce client used to create a base snapshot before overlaying local changes. Defaults to the value of <code>--p4client</code> if not specified.
<code>--p4changelist <change></code>	The changelist number (or default) whose changes are being tested and can be specified multiple times. If no changelists are specified, all client changelists will be used.
<code>--p4synctochange <change></code>	The changelist number the Preflight Job should use when sync'ing the source tree. Values are: <i>head</i> - The most recent changelist anywhere in the P4 depot (default). <i>have</i> - The changelist for the most recent file that was synced to 'p4client' changelist. This is a p4changelist number.

Subversion

The command-line options for the Subversion driver for `ecclientpreflight` are:

Subversion Options / Descriptions	
<code>--svnpath <path></code>	The path to the locally accessible source directory in which changes were made.
<code>--svnupdatetohead</code>	Use this option to update the agent workspace created during preflight -- updated to HEAD. By default, the agent workspace is updated to the revision found in the client workspace.
<code>--svnignoreexternals</code>	Causes the preflight process to ignore svn externals.

AccuRev

The command-line options for the AccuRev driver for `ecclientpreflight` are:

AccuRev Options / Descriptions	
<code>--accurevuser <user></code>	The value of ACCUREVUSER may be set in the environment also. If not specified, the default is to the ElectricFlow user.
<code>--accurevpasswd <password></code>	Required: The value of ACCUREVPASSWD may be set in the environment also. This is a required value.
<code>--accurevpending <pending></code>	Use this option to scan the client workspace for all pending elements using the "stat -fn -p" command. By default, the workspace is scanned for all kept elements using the "stat -fn -k" command.
<code>--accurevpath <path></code>	Required: The value of the ACCUREVPATH. Also may be set in the environment.

ClearCase

Preflight support for ClearCase is available only for snapshot views. The command-line options for the ClearCase driver for `ecclientpreflight` are:

ClearCase Options / Descriptions	
<code>--ccpath <path></code>	The path to the locally accessible source directory where changes were made.
<code>--ccUnixCSpecPath <unixCSpecPath></code>	The path to the appropriate config spec on a UNIX platform for the current view.
<code>--ccWinCSpecPath <winCSpecPath></code>	The path to the appropriate config spec on a Windows platform for the current view.
<code>--ccUnixRelativePath <unixRelativePath></code>	The relative path for the current view [on a UNIX platform] from the view root to the directory where changes were made.
<code>--ccWinRelativePath <winRelativePath></code>	The relative path for the current view [on a Windows platform] from the view root to the directory where changes were made.

Bazaar

The command-line options for the Bazaar driver for `ecclientpreflight` are:

Bazaar Options / Descriptions	
<code>--workdir <path></code>	The developer's source directory.
<code>--branch <name></code>	Branch name for the preflight.
<code>--method=<local remote></code>	<p><code>local</code> - get tracked and untracked changes in the current workingdir</p> <p><code>remote</code> - get changes between working tree and remote branch</p>

Git

The command-line options for the Git driver for `ecclientpreflight` are:

Git Options / Descriptions	
<code>--gitdir=dir</code>	The Git directory to process.
<code>--method=<local_all local_tracked remote></code>	<ul style="list-style-type: none">• <code>local_all</code> - get tracked and untracked changes between working tree and local repo• <code>local_tracked</code> - get tracked changes between working tree and local repo• <code>remote</code>

CVS

The command-line options for the CVS driver for `ecclientpreflight` are:

CVS Options / Descriptions	
<code>--cvsroot <path></code>	The path to the repository.
<code>--module <module></code>	The module name for the preflight.
<code>--workdir <path></code>	The developer's source directory.

Mercurial

The command-line options for the Mercurial driver for `ecclientpreflight` are:

Mercurial Options / Descriptions	
<code>--hgpath <path></code>	The path to the locally accessible source directory in which changes were made. Generally, this path is to the root of the workspace.

Repo

The command-line options for the Repo driver for `ecclientpreflight` are:

Repo Options / Descriptions	
<code>--repoworkdir <path></code>	The developer's source directory.
<code>--agentworkdir <path></code>	The path to the source directory, used by the agent.

StarTeam

The command-line options for the StarTeam driver for `ecclientpreflight` are:

StarTeam Options / Descriptions	
<code>--STProjectName <project></code>	The StarTeam project name.
<code>--workingdir <path></code>	Working dir for the updated files.

TFS

The command-line options for the Team Foundation Server driver for `ecclientpreflight` are:

TFS Options / Descriptions	
<code>--server <url></code>	The equivalent value to Collection in VS2008 and under. If you use this option, it is assumed you are using TF version VS2008 or under and it will use the option <code>/server</code> in the preflight. This field is required if you have VS2008.
<code>--collection <url></code>	The URL that points to <code>/collection</code> . This value is used for VS2010. If you specify this value, the command uses the option <code>/collection</code> when it executes the TF query for preflight. This field is required if you have VS2010.
<code>--localfolder</code>	The path to the locally accessible source directory in which changes were made. Generally, this is the path to the root of the workspace.
<code>--workspace <workspace></code>	The workspace containing the data for the preflight.

Vault

The command-line options for the Vault driver for `ecclientpreflight` are:

Vault Options / Descriptions	
<code>--workingdir <path></code>	Working dir for the updated files.

Troubleshooting

Client

The `--debug` option to `ecclientpreflight` provides additional information about internal actions taken when starting the preflight build. By modifying the properties containing driver scripts, debug output can be added where needed. If you want to capture debug information to a log file, use the `--log` option. By default, logs are stored in the user's home directory. To change this location, use the `--logDir` option.

Agent

The step log for the custom preflight step provides information about the snapshot being created and the files being overlayed. Debug output can be added where needed by modifying the properties containing the driver scripts

Properties

[Creating or modifying properties](#)

[Using property values](#)

[Property sheets and intrinsic properties](#)

[Property names and paths](#)

[Context-relative "shortcuts" to property paths](#)

[Property path shortcuts in ElectricFlow 5.0 and later on page 1012](#)

[Property name substitutions](#)

[Expandable properties](#)

[The property hierarchy](#)

[Special property references](#)

[Intrinsic properties listed by object type](#)

[ElectricFlow Object / Property tables](#)

Overview

ElectricFlow provides a powerful data model based around the notion of a *property*.

- A property is a string value with a name.
- Properties can have arbitrary names and values.
- You can attach properties to any object in the ElectricFlow system, such as a project, procedure, or job.
- After a property is created, it is stored in the ElectricFlow database.

- You can retrieve and modify the property value later.
- You can delete properties that you no longer need.

Properties are used extensively throughout the ElectricFlow system and provide a flexible and powerful mechanism to manage data about your builds.

Property Use Case Examples

- When a job starts, it computes a unique identifier for that build and saves it in a property. Later build steps can retrieve the property value to embed the build number in binaries generated during the job.
- When a build executes, it can set a property on the procedure to identify the source code version used for the build (for example, a tag or timestamp from your source code control system). The next time the build executes, it can retrieve the property value from the procedure and use it to extract information from your source code control system about the files that changed since the last run.
- Suppose you have a collection of machines for testing, and some machines have older test hardware and some machines have newer hardware with slightly different characteristics. You can set a property on each test machine resource to indicate which version of test hardware is available on which machine. Later, when a test executes, the test can retrieve the property for the machine where it ran and configure tests appropriately for that hardware. If you upgrade test hardware on a machine, all you need to do is change the property on that resource to reflect the new version.
- You can set job properties to indicate the build status produced by that job. For example, if your QA team finds fatal flaws in a build, it can mark the job accordingly. Builds that need to be preserved (release candidates or builds undergoing beta testing at customer sites) can be marked with properties so those builds are not deleted.
- When a job executes, properties are set for its job steps that hold metrics such as how many files compiled during a build step, how many tests executed during a test step, or how many errors were detected in the step. These property values are included in reports and can be examined later to compute trends over time. You can define additional properties for metrics useful to you.

ElectricFlow provides Intrinsic properties and allows you to create Custom properties. This Help topic enumerates properties available within ElectricFlow (Intrinsic properties), distinguishes between relative and absolute property paths, and describes property hierarchies.

Note: Intrinsic properties are case-sensitive. Custom properties, like all other object names in the ElectricFlow system, are "case-preserving," but *not case-sensitive*.

- **Intrinsic properties**

These properties represent attributes that describe the object to which they are attached, and are provided automatically by ElectricFlow for each similar type object. For example, every project has a `Description` property that can be referenced with a non-local property path such as `/projects/Examples/description`.

- **Custom properties**

Custom properties are identical to intrinsic properties and when placed on the same object, are referenced in the same manner, and behave in every way like an intrinsic object-level property with one exception: they are *not* created automatically when the object is created. Instead, custom properties can be added to objects already in the database before a job is started, or created dynamically by procedure steps during step execution.

Creating or modifying properties

You can set a property value by using one of the following three methods or applications: the automation platform web interface, the **ectool** command-line application, or the Perl API.

- Using the web interface, you will see a table labeled Custom Properties on the pages that display details for projects, procedures, and so on.
Click **Create New Property** to create a new property for that object, or click on an existing property to change its value.
- Using the ectool application, set a property value with a command like:

```
ectool setProperty /myJob/installStatus complete
```

This command sets the property named `/myJob/installStatus` to the value `complete`, and creates the property if it did not already exist (the meaning of property names like `/myJob/installStatus` is explained below). You can use ectool from within one job step to set properties accessed by later steps in the same job.

- Using the Perl API, you can use an external script or a script in a step with **ec-perl** as the shell. A Perl code example:

```
use ElectricCommander ();  
my $ec = new ElectricCommander;  
$ec->setProperty ("/myJob/installStatus", "complete", {jobStepId =>  
    $ENV{COMMANDER_JOBSTEPID})
```

Note: Using the Perl API may yield better performance if you are requesting multiple API calls in one step.

For more information, see the "Using the ElectricFlow API" topic in the ElectricFlow API Guide.

Using property values

A job step can access a property value by two methods. The first method is *substitution*, using the `$()` notation. Suppose you enter the following text as a command for a step:

```
make PLATFORM=$(platform)
```

Before running the step, ElectricFlow finds the property named `platform` and substitutes its value in the command string in place of `$(platform)`. For example, if the property contains the value "windows", the actual command executed is:

```
make PLATFORM=windows
```

The substitution method can be used for a command in a step and for any step fields, such as the resource or working directory. This method allows you to compute configuration information in an early step of a job, then use that configuration information to control later steps in the job.

Another way for a step to access the property value is with the ectool `getProperty` command. For example, the following command returns the property value named `platform`:

```
ectool getProperty platform
```

Property sheets and intrinsic properties

A property value can be a simple string or a nested *property sheet* containing its own properties. Property sheets can be nested to any depth, which allows you to create hierarchical collections of information.

Most objects have an associated "property sheet" that contains "custom properties" created by user scripts. The property sheet is an intrinsic property of the containing object called "property sheet", so to reference a project's custom property "branchName", you could specify `/projects/aProject/propertySheet/branchName`.

As a convenience, ElectricFlow allows the "property sheet" path element to be omitted and the path written as `/projects/aProject/branchName`. If there is no intrinsic property with the same name, the path will find the property on the property sheet.

Custom properties in a property sheet can be one of two types: *string property* or a *property sheet* property. String properties hold simple text values. Property sheet properties hold *nested* properties. Nested properties are accessed via the property sheet property of their containing sheet.

For example:

```
/projects/aProject/propertySheet/topSheet/propertySheet/propB
- or -
/projects/aProject/topSheet/propB
```

All information managed by ElectricFlow exists in the form of properties and property sheets and your own custom-created properties. For example, each project, procedure, and step is represented internally as a property sheet—the command for a step is actually a property associated with the step, and so on. Every value in the ElectricFlow system can be accessed as a property, using the naming facilities described below. These properties are called *intrinsic* properties. *ElectricFlow enforces some restrictions on intrinsic property values, whereas custom properties can have any value you choose.*

To learn more about intrinsic properties defined for an object by ElectricFlow, see the "Using the ElectricFlow Perl API" topic in the ElectricFlow API Guide. For example, to learn about intrinsic properties associated with each project, find the documentation for the `getProject` ectool command. The result of running this command is an XML document whose field names and values represent the properties for the project.

Note: At the end of this Help topic, find the current list of ElectricFlow intrinsic properties sorted by each object and its associated properties [in table format].

Property names and paths

Properties are named using multi-level paths such as `first/second/third`, which refers to a property named "third" in a property sheet named "second" in a property sheet named "first." ElectricFlow also supports an equivalent notation using brackets instead of slashes. In this format, slashes are not considered separators when they appear between brackets.

For example, `first[second]/third` and `first[second][third]` both refer to the same property as `first/second/third`. The bracket `[]` notation is based on matching brackets. For example, `steps[build[1]]` refers to a property named "build[1]" inside a property sheet named "steps" (it does not treat "build" as a property sheet containing a property named "1").

Property names/paths have two forms: *absolute* and *relative*.

Note: Some properties can be named by specifying the property name and a globally unique identifier for the property. For example, you can specify a job step using

```
/jobSteps/<jobStep ID>/
```

even though `jobSteps` is not a top-level (absolute) property name.

Absolute property paths

Absolute property paths are referenced by a fully-qualified path syntax that begins with a slash character ("/") followed by a top-level name. This path syntax is similar to a file system path specification. The first component after the "/" must be one of several reserved words that select a starting location to look up the property name. For example, consider the property name `/server/Electric Cloud/installDirectory`. `/server` means "lookup" starts in the topmost property sheet associated with the ElectricFlow server. This property name refers to the "installDirectory" property inside the server property sheet.

The system defines the following top-level names (absolute property names):

```
/applications/...
```

Start in the property sheet containing all applications. For example, `/applications[deploy]` refers to an application named "deploy" and `/applications[deploy]/processes[install]` refers to a process named "install" in that application.

```
/artifacts/...
```

Start in the property sheet containing all artifacts. For example, `/artifacts/myGrp:myKey/prop1` refers to the `prop1` property on the artifact whose name is "myGrp:myKey".

```
/artifactVersions/...
```

Start in the property sheet containing all artifact versions. For example, `/artifactVersions/myGrp:myKey:1.0-36/prop1` refers to the `prop1` property on the artifact version whose name is "myGrp:myKey:1.0-36".

Note: Throughout the API, you can substitute "groupId:artifactKey:version" wherever an `artifactVersionName` argument can be specified. This is the same if the `artifactVersionNameTemplate` specified in the artifact is in the form "groupId:artifactKey:version". When the template is different, it is convenient to be able to specify this tuple if you do not know the `artifactVersionName`.

```
/components/...
```

Start in the property sheet containing all components. For example, `/components[warfile]` refers to a component named "warfile" and `/component[warfile]/processes[backup]` refers to a component process named "backup" in that component.

```
/environments/...
```

Start in the property sheet containing all components. For example, `/environments[qeserver]` refers to an environment named "qeserver".

```
/groups/...
```

Start in the property sheet containing all groups. For example, `/groups[dev]` refers to the group named "dev".

/jobs/...

Start in the property sheet containing all jobs. For example, /jobs[eccloud.4096] refers to the job named "eccloud.4096." You can name a job using either its name (as in the preceding example) or using the unique identifier assigned to it by ElectricFlow.

/plugins/...

Start in the property sheet containing all plugins. For example, /plugins[EC-AgentManagement] refers to the currently promoted plugin named "EC-AgentManagement" and /plugins[EC-AgentManagement]/project/procedures[scpCopyFile] refers to a procedure named "scpCopyFile" in that plugin.

Note: There is a subtle difference between

```
ectool setProperty /plugins/EC-AgentManagement/project/foo 'bar'
```

and

```
ectool setProperty /plugins/EC-AgentManagement/foo 'bar'
```

The former places the property on the plugin's project and can be referenced by \$[/myProject/foo] while the latter places the property on the plugin object and will not be found via \$[/myPlugin/foo].

/projects/...

Start in the property sheet containing all projects. For example, /projects[nightly] refers to the project named "nightly," and /projects[nightly]/procedures[main] refers to a procedure named "main" in that project.

/repositories/...

Start in the property sheet containing all artifact repositories. For example, /repositories/repo1/prop1 refers to the prop1 property on the repository whose name is "repo1".

/resourcePools/...

Start in the property sheet containing all resource pools. For example, /resourcePools[linuxA] refers to the resource pool named "linuxA".

/resources/...

Start in the property sheet containing all resources. For example, /resources[linux1] refers to the resource named "linux1".

/server/...

Start in the top-level server property sheet. For example, /server/a refers to the server property named "a".

/users/...

Start in the property sheet containing all users. For example, /users[Bob] refers to the user named "Bob".

/workspaces/...

Start in the property sheet containing all workspaces. For example, `/workspaces[default]` refers to the workspace named "default."

Relative property paths

Property names that do not begin with `"/"` are *relative* and looked up starting in the *current context*. For example, when executing a job step, the current context includes properties defined for the job step, parameters for the current procedure, and global properties on the current job.

Relative property paths are distinguished from absolute property paths because they do not begin with one of the top-level names. To avoid having to construct full property paths, ElectricFlow supports the concept of a relative property path.

In use, the relative property path value is resolved by its context and a defined search order, which results in accessing the value of an absolute fully-specified property value. Contexts and search orders are as follows:

1. In a job step context, ElectricFlow searches for relative property paths in the following order:
 - a property on the job step object
 - a property of the parent job step object, which includes parameters of the procedure on which the step is defined
 - a property on the job object

Notes:

- The search can be enabled or disabled by using the `--extendedContextSearch` option on the `ectool getProperty` or `setProperty` commands. When searching for a property value, disable the search by setting the `--extendedContextSearch` switch to "false", requiring the property to exist on the job step object or return false. When writing a new value to a property, enable the search by setting the `--extendedContextSearch` switch to "true", allowing the search for a property within the search order before creating a new one. New properties are created on the job step object.
 - Parameters for subprocedure calls from job steps are searched for in a job step context.
 - For procedure parameters for nested subprocedures, properties referenced by parameters are looked up as described [above] for job steps.
2. When expanding schedule parameters, ElectricFlow searches the relative property path in the following order:
 - a property on the procedure being called
 - a property on the project on which the procedure being called is defined
 - a property on the server
 3. In a job name template context, ElectricFlow searches for the relative property path as a property on the job.
 4. In any other context, ElectricFlow searches for the relative property path as a property on the context object.

Context-relative "shortcuts" to property paths

A shortcut can be used to reference a property without knowing the exact name of the object that contains the property. You might think of a shortcut as another part of the property hierarchy. Shortcuts resolve to the correct property path even though its path elements may have changed because a project or procedure was renamed. Shortcuts are particularly useful if you do not know your exact location in the property hierarchical tree.

Shortcuts to property paths that provide convenient runtime access include:

`/myApplication/...`

Start in the property sheet for the `application` associated with the current job or job step.

`/myApplicationTier/...`

Start in the property sheet for the `applicationTier` associated with the current job step.

`/myArtifactVersion/...`

Start in the property sheet for the `artifactVersion` associated with the current context. The only context where this property has any value is in the `artifactVersionNameTemplate` field for the artifact.

`/myComponent/...`

Start in the property sheet for the `component` associated with the current context.

`/myCredential/...`

Start in the property sheet for the `credential` associated with the current job step. This form produces an error if the current job step does not have a credential.

`/myEvent/...`

This is a special property used only within an "email notifier." `/myEvent/` allows you to refer to fields associated with the notifier itself. You can include these properties, for example, in the text of the email you send out for notification:

`/myEvent/notifier` - This property contains the name of the notifier that created the notifier event. You created this name when you created the notifier.

`/myEvent/entity` - This property contains the object where the notifier is attached. Two possible values are "job" or "jobStep," depending on where the notifier is attached.

`/myEvent/source` - This property contains the name of either the job or the jobStep where the notifier is attached.

`/myEvent/type` - This property defines whether the notifier is an "On Start" or an "On Completion" notifier. Two possible values are "STARTED" or "COMPLETED".

`/myEvent/time` - This property contains the time when the notifier occurred. The time is always specified in GMT and uses a formatted string, for example,
2009-06-11T21:00:56.502Z

`/myEvent/timeMillis` - This property contains the "timestamp" in milliseconds when the notifier occurred. This value is the number of milliseconds since January 1, 1970 GMT. The `timeMillis` corresponding to the time in the previous example is: 1244754056502

`/myEnvironment/...`

Start in the property sheet for the `environment` associated with the current job or job step.

`/myEnvironmentTier/...`

Start in the property sheet for the `environmentTier` associated with the current job step.

`/myJob/...`

Start in the global property sheet for the current job. `/myJob/` points directly to the job object so you can reference built-in job properties (`jobName`, `createTime`, `outcome`, and so on) and custom properties. Also, this property sheet can be used to hold working data that needs to be passed from one step to another within the job.

`/myJobStep/...`

Start in the property sheet for the current job step.

`/myParent/...`

Start in the global property sheet for the parent job step or job if this is a top-level step. `/myParent/` points directly to the job or job step object, so you can reference intrinsic or custom properties. For example, you can reference the parameters that were passed to this procedure.

In addition, you can reference a sibling step by calling `/myParent/jobSteps/<sibling step name>`.

Or, you can create additional properties on `/myParent/` to pass information from one step in the procedure to another:

`step1` calls `setProperty /myParent/results 100`

`step2` can call `getProperty /myParent/results`

`/myParent/`.

`/myProcedure/...`

Start in the property sheet for the procedure in which the current job step was defined. If the current job step is executing as part of a nested procedure, `/myProcedure/` refers to the innermost nested procedure.

`/myProcess/...`

Start in the property sheet for the application or component process in which the current job or job step was defined.

`/myProcessStep/...`

Start in the property sheet for the application or component process step in which the current job step was defined.

`/myProject/...`

Start in the property sheet for the project in which the current job step was defined. If the job step has nested procedure invocations, this is the project associated with the innermost nested procedure, for example, the project associated with `/myProcedure/`.

`/myResource/...`

Start in the property sheet for the resource assigned to the current job step.

`/myResourcePool/...`

Start in the property sheet for the resource pool that provided the resource for the current job step. Returns null if the step did not specify a resource pool.

`/myStep/...`

Start in the property sheet for the current step. "Step" refers to the (static) definition of a step, which is part of a procedure— this is different from a job step, which represents a step when it executes (dynamically) in a job. Use `/myJobStep/` to access the job step.

`/myState/...`

Start in the property sheet for the state object so you can reference built-in and custom state properties or find parameter values passed to that state. When accessed from a state, `/myState/` refers to that state. When accessed from a transition, `/myState` refers to the transition's owning state. When accessed from a job or job step, `/myState/` refers to the state that launched that job as a subjob.

`/mySubjob/...`

Start in the property sheet for the subjob so you can reference built-in and custom job properties, parameters passed to the job, and properties on steps within that job. When accessed from a transition, `/mySubjob/` refers to the subjob started by the state that owns that transition. This property path is particularly useful in conditions for On Completion transitions because the outcome or other information for the subjob can influence which state the workflow transitions to next.

`/mySubworkflow/...`

Start in the property sheet for the subworkflow so you can reference built-in and custom workflow properties. When accessed from a transition, `/mySubworkflow/` refers to the subworkflow started by the state that owns that transition. This property path is particularly useful in conditions for On Completion transitions because the active state and other information for the subworkflow can influence which state the workflow transitions to next. You can access information about states and transitions belonging to the workflow by using the path `/mySubworkflow/states/someState` or `/mySubworkflow/states/someState/transitions/someTransition`.

`/myTransition/...`

Start in the property sheet for the transition object so you can reference intrinsic and custom transition properties. `/myTransition/` is accessible only from a transition.

`/myUser/...`

This property can be used only if the current session is associated with: the predefined "admin" user, a user defined as "local", or a user defined by a Directory Provider (LDAP or ActiveDirectory). This

property cannot be used if the user is a "project principal", which is normally the case when running inside an ElectricFlow step.

For example, with an interactive login you can use:

```
ectool getProperty /myUser/userName (to get the user name of the logged in user, or...)
```

```
ectool getProperty /myUser/email (to get the email address)
```

```
/myWorkflow/...
```

Start in the property sheet for the workflow object so you can reference built-in and custom workflow properties. When accessed from a state or transition, `/myWorkflow/` refers to the "owning" workflow. When accessed from a job or job step, `/myWorkflow/` refers to the workflow whose state launched that job as a subjob. You can access information about states and transitions belonging to the workflow by using the path `/myWorkflow/states/someState` or `/myWorkflow/states/someState/transitions/someTransition`.

```
/myWorkflowDefinition/...
```

Starts in the property sheet for the workflow definition object so you can reference built-in and custom workflow properties.

```
/myWorkspace/...
```

Start in the property sheet for the workspace associated with the current job step.

Property path shortcuts in ElectricFlow 5.0 and later

The following shortcuts to property paths are available in ElectricFlow 5.0 and later. For more information, go to [Context-relative "shortcuts" to property paths](#) on page 1009.

Shortcuts	Descriptions	For jobs	For job steps
<code>/myApplication/...</code>	Starts in the property sheet for the application associated with the current job or job step.	Yes	Yes
<code>/myApplicationTier/...</code>	Starts in the property sheet for the application tier associated with the current job step.	No	Yes
<code>/myComponent/...</code>	Starts with the property sheet for the component associated with the current job step.	No	Yes
<code>/myCredential/...</code>	Starts with the property sheet for the credential associated with the current job step.	No	Yes
<code>/myEnvironment/...</code>	Starts in the property sheet for the environment associated with the current job or job step.	Yes	Yes

Shortcuts	Descriptions	For jobs	For job steps
/myEnvironmentTier/...	Starts in the property sheet for the environment tier associated with the current job step.	No	Yes
/myJob/...	Starts in the property sheet for the job associated with the current job or job step.	Yes	Yes
/myJobStep/...	Starts in the property sheet for the job step associated with the current job step.	No	Yes
/myParent/...	Starts in the global property sheet for the parent job step.	No	Yes
/myPlugin/...	Starts in the property sheet for the plugin associated with the current job.	Yes	No
/myProcedure/...	Starts in the property sheet for the procedure in which the current job or job step was defined.	Yes	Yes
/myProcess/...	Starts in the property sheet for the process in which the current job or job step was defined.	Yes	Yes
/myProcessStep/...	Starts in the property sheet for the process step in which the current job step was defined.	No	Yes
/myProject/...	Starts in the property sheet for the project in which the current job was defined.	Yes	No
/myResource/...	Starts in the property sheet for the resource associated with the current job step.	No	Yes
/myResourcePool/...	Starts in the property sheet for the resource pool associated with the current job step.	No	Yes
/myState/...	Starts in the property sheet for the state object so you can reference built-in and custom state properties or find parameter values passed to that state.	Yes	Yes
/myStep/...	Starts in the property sheet for the step associated with the current job step.	No	Yes

Shortcuts	Descriptions	For jobs	For job steps
<code>/myWorkflow/...</code>	Starts in the property sheet for the workflow object so you can reference built-in and custom workflow properties.	Yes	Yes
<code>/myWorkflowDefinition/...</code>	Starts in the property sheet for the workflow definition object so you can reference built-in and custom workflow properties.	Yes	Yes
<code>/myWorkspace/...</code>	Starts in the property sheet for the workspace associated with the current job step.	No	Yes

Property name substitutions

Property names can contain references to other properties, which are then substituted into the property name before looking it up. For example, consider the following property name:

```
/myStep/${/myProcedure/name}
```

If the value of `/myProcedure/name` is "xyz", the property above is equivalent to `/myStep/xyz`.

Expandable properties

Property values can contain property references using the `"${}"` notation.

For example:

1. Create a property named "foo" with a value of `hello ${bar}`.
2. Create a property named "bar" with a value of `world`.
3. Reference "foo" (either using `${foo}` or `ectool getProperty foo`). The value "hello world" is returned.

If you want just the literal value of "foo" (useful in the UI, for example), you can use the `expand` option in `ectool`:

```
ectool getProperty foo --expand false. The value "hello ${bar}" will be returned.
```

Properties are expanded by default when you use `getProperty` or `getProperties`.

If the value of a property contains `"${}"` but you do not want it to be interpreted as a property reference, you can use the `expandable` option:

```
ectool setProperty symbols '${!@}#' --expandable false.
```

This option can be toggled in the web UI as well. Properties are expandable by default.

Because you cannot control where your expandable property might be referenced (and therefore which context is used during expansion), Electric Cloud recommends using absolute paths when referencing a property from the value of another property.

In the example above, if you define both "foo" and "bar" as properties on a project "proj1", you might assume there is no problem with the value of "foo". However, if you later reference "foo" from a job under "proj1" (for example, `$/myProject/foo`), foo will be referenced with the job step as its context. Therefore, when the value of "foo" is expanded, you will get a `PROPERTY_REFERENCE_ERROR` because "bar" is not defined in the context of the job step.

Custom property names and values

The following properties are used by the standard ElectricFlowUI, so you should use these property names whenever possible, and avoid using these names in ways that conflict with the definitions below.

- **preSummary** - if this property exists, its value is displayed in the "Status" field (on the Job Details page) for this step. This property appears *before* whatever would normally be displayed for status. If the property contains multiple lines separated by newline characters, each line is displayed on a separate line in the status field.
- **postSummary** - if this property exists, its value is displayed in the "Status" field (on the Job Details page) for this step. This property appears *after* whatever would normally be displayed for status. If the property contains multiple lines separated by newline characters, each line is displayed on a separate line in the status field.
- **summary** - if this property exists, its value is displayed in the "Status" field (in the job reports) for this step, replacing whatever would normally be displayed for status. If the property contains multiple lines separated by newline characters, each line is displayed on a separate line in the status field.

The property hierarchy

All ElectricFlow properties fall into a hierarchical structure, and you can reference any property using an absolute path from the root of the hierarchy. This includes properties you define and intrinsic properties defined by ElectricFlow.

For example, each step contains a property "resource" that provides the resource name to use for that step. All steps of a procedure exist as property sheets underneath the procedure. All procedures in a project exist as property sheets underneath the project, and so on.

The examples below illustrate some nesting relationships between objects.

For example, the notation "*project/procedures[procedureName]*" means each project object contains a property sheet named "procedures" holding all procedures defined within that project. Within the procedures property sheet, there is a nested property sheet for each procedure, named after the procedure. Thus, the name "*projects[a]/procedures[b]*" refers to a procedure named "b" contained in a project named "a."

- Each project contains procedures, schedules, credential definitions, workflow definitions, and workflows:

```
project/procedures[procedureName]
project/schedules[scheduleName]
project/credentials[credentialName]
project/workflowDefinitions[workflowName]
project/workflows[workflowName]
```

- Each procedure contains steps and parameters. The parameters are "formal parameters," meaning they specify parameter names the procedure will accept, whether each parameter is required, and so on:
procedure/steps[stepName]
procedure/formalParameters[paramenterName]
- Steps that invoke subprocedures contain parameter values for the subprocedure. These are called "actual parameters" because they provide actual values that will be passed into the subprocedure:
step/actualParameters[parameterName]
- Each job contains a collection of job step objects for steps in the outermost procedure, along with parameter values passed into the job when it was invoked:
job/jobSteps[stepName]
job/actualParameters[parameterName]
- If a job step invokes a nested subprocedure, its property sheet contains parameter values that were passed into the nested subprocedure, plus all job steps corresponding to that procedure:
jobStep/actualParameters[parameterName]
jobStep/jobSteps[stepName]
- Schedules contain parameter values for the procedures they invoke. These are called "actual parameters" because they provide actual values passed into the procedure:
schedule/actualParameters[parameterName]
- Workflow definitions contain state definitions:
workflowDefinition/stateDefinitions[stateDefinitionName]
- Transition definitions contain actual parameters:
transitionDefinition/actualParameters[parameterName]
- State definitions contain transition definitions, actual parameters, and formal parameters:
stateDefinition/actualParameters[parameterName]
stateDefinition/formalParameters[parameterName]
stateDefinition/transitionDefinitions[transitionDefinitionName]
- Workflows contains states:
workflow/states[stateName]
- Transitions contain actual parameters:
transition/actualParameters[parameterName]
- States contain transitions, actual parameters, and formal parameters:
state/actualParameters[parameterName]
state/formalParameters[parameterName]
state/transitions[transitionName]

Special property references

ElectricFlow also supports several special property reference forms that are described in the following subsections.

increment

Use this form to increment the value of an integer property before returning its value. For example, suppose property xyz has the value 43. The property reference `$/increment xyz` first increments the value of property xyz to 44, then returns 44.

timestamp

Use this form to generate a formatted timestamp value. For example, the property reference `$/timestamp yyyy-MMM-d hh:mm` returns the current time in a form such as "2007-Jun-19 04:36". The pattern following `/timestamp` specifies how to format the time and defaults to "yyyyMMddHHmm". The pattern follows conventions for the Java class `SimpleDateFormat`, where various letters are substituted with various current time elements. For more information, go to <http://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html> for `SimpleDateFormat` class documentation.

Here are some of the supported substitutions:

- y - Year, such as "2007" or "07"
- M - Month, such as "April", "Apr", or "04"
- w - Week in year, such as "27"
- W - Week in month, such as "3"
- D - Day in year, such as "194"
- d - Day in month, such as "18"
- E - Day in the week, such as "Tuesday" or "Tue"
- a - am/pm marker, such as "PM"
- H - Hour in day (0-23), such as "7"
- h - Hour in am/pm (1-12), such as "11"
- m - Minute in hour, such as "30"
- s - Second in minute, such as "15"
- S - Millisecond, such as "908"
- z - Time zone such as "Pacific Standard Time", "PST", or "GMT-08:00"
- Z - Time zone such as "-0800"

Repeated letters cause longer forms to be substituted. For example, "yy" substitutes the last 2 digits of the current year, whereas "yyyy" substitutes the 4-digit year number. Single quotes can be used to substitute text directly without the above interpretations.

javascript

This form executes a Javascript code fragment inside the ElectricFlow server and returns the result computed by that code.

For example, `$/javascript 4*2` returns "8.0". Javascript code can be arbitrarily long and include multiple statements. The value of the last statement is returned by the property reference.

Javascript code executes in an interpreter that provides access to ElectricFlow properties:

1. The normal way to access property values is through Javascript objects. Global Javascript objects named "server," "projects," and so on, exist and correspond to all top-level objects in an absolute property path.

For example, there is a Javascript object named `server` that corresponds to the path `/server`, and a Javascript object named `myJob` that corresponds to the path `/myJob`. You can use either `"."` or `"["` notation to access properties within the object.

For a more complete list of top-level objects, see the ["Absolute property paths"](#) section.

Examples:

```
$[/javascript myJob.mightExist]
this is a safe way to refer to an optional parameter
```

```
$[/javascript (myJobStep.errors > 0) ? "step failed" : "no errors"]
test a value and return another string based on the result
```

```
$[/javascript server.settings.ipAddress]
refer to a property in a nested property sheet
```

```
$[/javascript server["Electric Cloud"].installDirectory]
if the property name has a space, use [""] notation
```

```
$[/javascript server["Electric Cloud"]["installDirectory"]]
do not use the "." in front of the "["
```

2. You can also call the function, `"getProperty"`. It takes a property name as argument and returns the value of that property. The case where this is most useful is when you want to access another special property reference.

Examples:

```
$[/javascript getProperty ("/timestamp yyyy-MMM-d hh:mm")]
returns the current time
```

```
$[/javascript getProperty ("/increment /myJob/jsCount")]
this is the only way to change a property via javascript
```

For example, consider the following property reference:

```
$[/javascript (getProperty("/myJobStep/errors") > 0) ? "step failed" : "no errors"]
```

This example returns "step failed" if the property "errors" on the current job step had a value greater than zero, and it returns "no errors" otherwise.

3. If calling the `"setProperty"` function, similar to `getProperty`, there are two variations on the function:
 - A *global* function variation that uses the current context object.
The global function takes 2 or 3 arguments. The 3-argument version takes a context object, a path, and a value. The 2-argument version omits the context object and uses the current context object (for

example, *job step*).

Example:

```
setProperty (myProject, "foo", "bar")
```

would set the value of the "foo" property on the current project to "bar".

- An *object* function variation that can be called on objects, which uses that object as a context object. The object function takes two arguments: a "path" and a "value".

Example:

```
server.setProperty ("foo", "bar")
```

would set the "foo" property on the server object to the value "var".

ElectricFlow Intrinsic Properties listed by object type

Each of the ElectricFlow object types [in the list below] links to a table for that object.

Each table includes a list of intrinsic properties applicable to that object and provides the property type and description.

acl	repository
artifact	resource
artifactVersion	resourcePool
credential	resourceUsage
directoryProvider	schedule
emailConfig	server
emailNotifier	state
formalParameter	stateDefinition
gateways	step
group	transition
job	transitionDefinition
jobStep	user
logEntry	workflow
procedure	workflowDefinition
project	workspace
property	zones
propertySheet	

ElectricFlow Property Type definitions

This table provides Property Type definitions for each property listed in the following tables.
(A Property Type precedes each property description in the Description column.)

Type	Definition
boolean	One of two possible values - either <i>true</i> or <i>false</i> . These values are more frequently represented by the numbers "0" and "1", where "0" equals false and "1" equals true.
date	A millisecond precision UTC date in ISO 8601 form: [YYYY] - [MM] - [DD] T [hh] : [mm] Z For example, 2007-06-19T04:36:22.000Z
id	Each time an object is created, ElectricFlow generates a unique ID number for that object.
name	This is a unicode string value with a maximum of 255 characters.
number	This is a simple integer numeric value.
reference	This property refers to another object.
string	This is a unicode string value with a maximum CLOB size of the database, but only the first 450 characters are indexed, which means a defined search will not "see" beyond the first 450 characters.

ElectricFlow Object / Property tables

In the following tables, the Description column displays the property type preceding the property description.

Object Type: <code>acl</code> Description: An <i>acl</i> is an Access Control List.																													
Property Name	Description																												
<code>aclId</code>	<code>id</code> : The unique identifier for this <code>acl</code> object. Other objects can refer to this <code>acl</code> by its ID.																												
<code>inheriting</code>	<code>boolean</code> : If true, the ACL inherits ACEs from the ACL's parent.																												
<code>ownerType</code>	<p>This is any type of object the ACL controls and can be any of the objects listed below.</p> <table> <tr> <td><code>ace</code></td><td><code>project</code></td></tr> <tr> <td><code>acl</code></td><td><code>property</code></td></tr> <tr> <td><code>admin</code></td><td><code>propertySheet</code></td></tr> <tr> <td><code>artifact</code></td><td><code>repository</code></td></tr> <tr> <td><code>artifactVersion</code></td><td><code>resource</code></td></tr> <tr> <td><code>event</code></td><td><code>resourcePool</code></td></tr> <tr> <td><code>formalParameter</code></td><td><code>server</code></td></tr> <tr> <td><code>group</code></td><td><code>schedule</code></td></tr> <tr> <td><code>job</code></td><td><code>systemObject</code></td></tr> <tr> <td><code>jobStep</code></td><td><code>user</code></td></tr> <tr> <td><code>license</code></td><td><code>userSettings</code></td></tr> <tr> <td><code>notifier</code></td><td><code>workspace</code></td></tr> <tr> <td><code>procedure</code></td><td><code>plugin</code></td></tr> <tr> <td><code>procedureStep</code></td><td></td></tr> </table>	<code>ace</code>	<code>project</code>	<code>acl</code>	<code>property</code>	<code>admin</code>	<code>propertySheet</code>	<code>artifact</code>	<code>repository</code>	<code>artifactVersion</code>	<code>resource</code>	<code>event</code>	<code>resourcePool</code>	<code>formalParameter</code>	<code>server</code>	<code>group</code>	<code>schedule</code>	<code>job</code>	<code>systemObject</code>	<code>jobStep</code>	<code>user</code>	<code>license</code>	<code>userSettings</code>	<code>notifier</code>	<code>workspace</code>	<code>procedure</code>	<code>plugin</code>	<code>procedureStep</code>	
<code>ace</code>	<code>project</code>																												
<code>acl</code>	<code>property</code>																												
<code>admin</code>	<code>propertySheet</code>																												
<code>artifact</code>	<code>repository</code>																												
<code>artifactVersion</code>	<code>resource</code>																												
<code>event</code>	<code>resourcePool</code>																												
<code>formalParameter</code>	<code>server</code>																												
<code>group</code>	<code>schedule</code>																												
<code>job</code>	<code>systemObject</code>																												
<code>jobStep</code>	<code>user</code>																												
<code>license</code>	<code>userSettings</code>																												
<code>notifier</code>	<code>workspace</code>																												
<code>procedure</code>	<code>plugin</code>																												
<code>procedureStep</code>																													
<code>parentId</code>	<code>id</code> : The parent ACL.																												

[\[back to top\]](#)

Object Type: artifact

Description: An *artifact* is an object that contains zero or more artifact versions.

An artifact has two purposes:

1. To group artifact versions and provide a template for naming the versions
2. To restrict who can publish artifact versions, based on `groupId:artifactKey`

Property Name	Description
<code>acl</code>	<code>reference: acl</code>
<code>artifactId</code>	<code>id</code> : The artifact's ID number.
<code>artifactKey</code>	<code>string</code> : User-specified identifier for this artifact. This field may consist of alphanumeric characters, spaces, underscores, hyphens, and periods.
<code>artifactName</code>	<code>name</code> : The name of this artifact.
<code>artifactVersionNameTemplate</code>	<code>name</code> : The template for artifact version names published to this artifact.
<code>createTime</code>	<code>date</code> : The time when this object was created.
<code>description</code>	<code>string</code> : A user-specified text description of the object.
<code>groupId</code>	<code>id</code> : A user-generated group name for this artifact. This field may consist of alphanumeric characters, spaces, underscores, hyphens, and periods.
<code>lastModifiedBy</code>	<code>name</code> : This shows who (generally, a user name) last modified this object.
<code>modifyTime</code>	<code>date</code> : The time when this object was last modified.
<code>owner</code>	<code>name</code> : The person (user name) who created the object.
<code>propertySheet</code>	<code>reference: propertySheet</code>

[\[back to top\]](#)

Object Type: artifactVersion

Description: An artifact version is an object that represents a user-defined unit of related files typically produced by one job and consumed by one or more other jobs.

Property Name	Description
acl	<code>reference: acl</code>
artifactKey	<code>string</code> : User-specified identifier for this artifact. This field may consist of alphanumeric characters, spaces, underscores, hyphens, and periods.
artifactName	<code>name</code> : The name of the artifact.
artifactVersionId	<code>id</code> : The ElectricFlow-generated ID number for this artifact version.
artifactVersionName	<code>name</code> : The name of the artifact version.
artifactVersionState	<code>string</code> : Possible values are: available publishing unavailable
buildNumber	<code>number</code> : User-defined build number component of the version attribute for the artifact version.
createTime	<code>date</code> : The time when this object was created.
description	<code>string</code> : A user-specified text description of the object.
groupId	<code>id</code> : A user-generated group name for this artifact. This field may consist of alphanumeric characters, spaces, underscores, hyphens, and periods.
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
majorMinorPatch	<code>string</code> : <code>major.minor.patch</code> component of the version attribute for the artifact.
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.

Object Type: artifactVersion

Description: An artifact version is an object that represents a user-defined unit of related files typically produced by one job and consumed by one or more other jobs.

Property Name	Description
propertySheet	<code>reference</code> : <code>propertySheet</code>
publisherJobId	<code>id</code> : The ElectricFlow-generated ID number for the job that published the artifact version.
publisherJobName	<code>string</code> : The name of the job that published the artifact version.
publisherJobStepId	<code>id</code> : The ElectricFlow-generated ID number for the job step that published the artifact version.
qualifier	<code>string</code> : User-defined qualifier component of the version attribute for the artifact.
repositoryName	<code>name</code> : The name of the artifact repository.
version	<code>string</code> : An artifact version specification uses the following form: <i>major.minor.patch.qualifier.buildNumber</i>

[\[back to top\]](#)

Object Type: credential

Description: In ElectricFlow, a *credential* is an object that stores a username and password for later use.

Property Name	Description
acl	<code>reference</code> : <code>acl</code>
createTime	<code>date</code> : The time when this object was created.
credentialId	<code>id</code> : The credential's ID number.
credentialName	<code>name</code> : The name of this credential.

Object Type: credential

Description: In ElectricFlow, a *credential* is an object that stores a username and password for later use.

Property Name	Description
description	<code>string</code> : A user-specified text description of the object.
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.
password	<code>string</code> : The password corresponding to the user name and this credential.
projectName	<code>name</code> : The name of the <code>project</code> that contains this credential.
propertySheet	<code>reference</code> : <code>propertySheet</code>
userName	<code>name</code> : A saved string that represents the name portion of a credential, typically a user account name.

[\[back to top\]](#)

Object Type: directoryProvider

Description: A `directoryProvider` is an object containing information about an external directory service (LDAP or ActiveDirectory).

Property Name	Description
acl	<code>reference</code> : <code>acl</code>
commonGroupNameAttribute	<code>string</code> : The attribute in a group record that contains the common group name. If specified, this name is used only when searching for groups from an external provider.

Object Type: directoryProvider

Description: A directoryProvider is an object containing information about an external directory service (LDAP or ActiveDirectory).

Property Name	Description
createTime	<code>date</code> : The time when this object was created.
description	<code>string</code> : A user-specified text description of the object.
directoryProviderId	<code>id</code> : The ID of the directory provider.
domainName	<code>string</code> : The domain name from which Active Directory servers are automatically discovered.
emailAttribute	<code>string</code> : The attribute in a user record that contains the user's email address. If the attribute was not specified, the account name and domain name are concatenated to form an email address.
enableGroups	<code>boolean</code> : Determines whether or not external groups are enabled for the directory provider. Defaults to "true".
fullUserNameAttribute	<code>name</code> : The attribute in a user record that contains the user's full name (first and last) for display in the UI.
groupBase	<code>string</code> : This string is prepended to the <code>basedn</code> to construct the directory DN that contains group records.
groupMemberAttributes	<code>string</code> : A comma-separated attribute name list that identifies a group member.
groupMemberFilter	<code>string</code> : Two common forms of group record in LDAP directories: POSIX style groups where members are identified by account name, and <code>groupOfNames</code> or <code>uniqueGroupOfNames</code> records where members are identified by the full user DN. Both forms are supported, so the query is passed to parameters: "{0}" is replaced with the full user record DN, and "{1}" is replaced with the user's account name.
groupNameAttribute	<code>name</code> : The group record that contains the name of the group.

Object Type: directoryProvider

Description: A directoryProvider is an object containing information about an external directory service (LDAP or ActiveDirectory).

Property Name	Description
groupSearchFilter	string: This LDAP query is performed in the context of the groups directory to enumerate group records.
lastModifiedBy	name: This shows who (generally, a user name) last modified this object.
managerDn	name: The DN of a user who has read-only access to LDAP user and group directories.
modifyTime	date: The time when this object was last modified.
owner	name: The person (user name) who created the object.
propertySheet	reference: propertySheet
providerIndex	number: The index that specifies the search order across multiple directory providers. For example: 2 LDAP providers, one with index "0" and one with index "1" means the providers will be searched in that numerical order.
providerName	name: This human-readable name will be displayed in the user interface to identify users and groups that come from this provider.
providerType	string: <ldap activedirectory>
realm	string: An identifier (string) used for LDAP directory providers so users and groups (within LDAP) can be uniquely identified in "same name" collisions across multiple directory providers. The realm is appended to the user or group name when stored in the ElectricFlow server. For example, <user>@dir (where the realm is set to "dir").
url	string: The server URL is in the form <code>protocol://host:port/basedn</code> . Protocol is either ldap or ldaps (for secure LDAP).

Object Type: directoryProvider

Description: A directoryProvider is an object containing information about an external directory service (LDAP or ActiveDirectory).

Property Name	Description
userBase	string: This string is prepended to the <code>basedn</code> to construct the directory DN that contains user records.
userNameAttribute	name: The attribute in a user record that contains the user's account name.
userSearchFilter	string: This LDAP query is performed in the context of the user directory to search for a user by account name. The string "{0}" is replaced with the user's login ID. Typically, the query compares a user record attribute with the substituted user login ID.
userSearchSubtree	boolean: If true, the subtree below the user base is searched recursively.
useSSL	boolean: This flag is used to specify SSL to communicate with your Active Directory servers. Note: Transport Layer Security (TLS) has replaced Secure Sockets Layer version 3.0 (SSLv3) on the ElectricFlow web server and the ElectricFlow server.

[\[back to top\]](#)

Object Type: emailConfig

Description: An *emailConfig* is an object that stores information created and used to communicate with the email server.

Property Name	Description
acl	reference: <code>acl</code>
configName	string: The name of the email configuration.

Object Type: emailConfig Description: An <i>emailConfig</i> is an object that stores information created and used to communicate with the email server.	
Property Name	Description
createTime	date: The time when this object was created.
description	string: A user-specified text description of the object.
emailConfigId	id: The ElectricFlow-generated ID for the email configuration.
emailConfigName	string: The name of the email configuration.
lastModifiedBy	name: This shows who (generally, a user name) last modified this object.
mailFrom	string: The email address used as the email sender address for notifications.
mailHost	string: The name of the email server host.
mailPort	number: The port number for the mail server. The protocol software determines the default value (25 for SMTP and 465 for SSMTP).
mailProtocol	string: This is either SSMTP or SMTP (not case sensitive). Default is SMTP.
mailUser	name: An individual or a generic name like "ElectricFlow" -- the name of the email user on whose behalf ElectricFlow sends email notifications.
modifyTime	date: The time when this object was last modified.
owner	name: The person (user name) who created the object.
propertySheet	reference: propertySheet

[\[back to top\]](#)

Object Type: emailNotifier

Description: An *emailNotifier* is an object that stores information created and used to notify users about various types information, including status.

Property Name	Description
acl	<code>reference:</code> <code>acl</code>
condition	<code>string</code> : Only send mail if the condition evaluates to "true". The condition is a string subject to property expansion. The notification will NOT be sent if the expanded string is "false" or "0". If no condition is specified, the notification is ALWAYS sent.
configName	<code>string</code> : The name of the email configuration.
container	<code>id</code> : The object ID to which the email notifier is attached (for example: procedure-123).
containerType	<code>string</code> : The type of object that the notifier is attached to (procedure, step, job, jobstep).
createTime	<code>date</code> : The time when this object was created.
description	<code>string</code> : A user-specified text description of the object.
destinations	<code>string</code> : A space-separated list of valid email addresses, email aliases, or ElectricFlow user names, or a string subject to property expansion that expands into such a list.
emailNotifierId	<code>id</code> : The ElectricFlow-generated ID for the email notifier.
eventType	<code>string</code> : <code><onEnter onStart onCompletion></code> <code>onStart</code> triggers an email notification when the job or job step begins. <code>onCompletion</code> , default, triggers an email notification when the job finishes, no matter how it finishes.
formattingTemplate	<code>string</code> : The email address used as the email sender address for notifications.

Object Type: emailNotifier

Description: An *emailNotifier* is an object that stores information created and used to notify users about various types information, including status.

Property Name	Description
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
modifyTime	<code>date</code> : The time when this object was last modified.
notifierName	<code>name</code> : The name of the email notifier.
owner	<code>name</code> : The person (user name) who created the object.
propertySheet	<code>reference</code> : <code>propertySheet</code>

[\[back to top\]](#)

Object Type: formalParameter

Description: A *formalParameter* is a parameter expected by a procedure, including its name, a default value, and an indication of whether the parameter is required. Formal parameters are different from "actual parameters"--- formal parameters define the type of parameters a procedure is expecting, and actual parameters provide values to use at run-time.

Property Name	Description
container	<code>string</code> : An object ID for a "container" that contains formal parameters.
containerType	<code>string</code> : The type of object containing the formal parameter.
createTime	<code>date</code> : The time when this object was created.
defaultValue	<code>string</code> : The <code>formalParameter</code> 's default value.

Object Type: `formalParameter`

Description: A *formalParameter* is a parameter expected by a procedure, including its name, a default value, and an indication of whether the parameter is required. Formal parameters are different from "actual parameters"--- formal parameters define the type of parameters a procedure is expecting, and actual parameters provide values to use at run-time.

Property Name	Description
<code>description</code>	<code>string</code> : A user-specified text description of the object.
<code>expansionDeferred</code>	<code>boolean</code> : Determines whether or not the property expansion is deferred.
<code>formalParameterId</code>	<code>id</code> : This is this formal parameter's ID.
<code>formalParameterName</code>	<code>name</code> : This is this formal parameter's name.
<code>lastModifiedBy</code>	<code>name</code> : This shows who (generally, a user name) last modified this object.
<code>modifyTime</code>	<code>date</code> : The time when this object was last modified.
<code>owner</code>	<code>name</code> : The person (user name) who created the object.
<code>required</code>	<code>boolean</code> : If true, a value for this parameter must be supplied when the procedure is called.
<code>type</code>	<code>name</code> : The custom type of <code>formalParameter</code> .

[\[back to top\]](#)

Object Type: gateway

Description: To communicate with a resource, workspace, or artifact repository server in another zone, a "gateway" must be created. A gateway object contains two resource (agent) machines, for example, GatewayResource1 and GatewayResource2—each configured to communicate with the other. One gateway resource resides in the *source* zone and the other in the *target* zone. A gateway is bidirectional and informs the ElectricFlow server that each gateway machine is configured to communicate with its other gateway machine (in another zone).

Property Name	Description
acl	<code>reference</code> : acl
createTime	<code>date</code> : The time when this object was created.
description	<code>string</code> : A user-specified text description of the object.
gatewayDisabled	<code>boolean</code> : If set to 1 (true), the gateway is disabled.
gatewayId	<code>id</code> : The unique ElectricFlow-generated ID for this gateway.
gatewayName	<code>string</code> : The name of the gateway.
hostName1	<code>string</code> : The agent host name where <i>Resource1</i> resides. This host name is used by <i>Resource2</i> to communicate with <i>Resource1</i> . Do not specify this option if you want to use the host name from <i>Resource1</i> 's definition.
hostName2	<code>string</code> : The agent host name where <i>Resource2</i> resides. This host name is used by <i>Resource1</i> to communicate with <i>Resource2</i> . Do not specify this option if you want to use the host name from <i>Resource2</i> 's definition.
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.
port1	<code>number</code> : The port number used by <i>Resource1</i> - defaults to the port number used by the resource.

Object Type: gateway

Description: To communicate with a resource, workspace, or artifact repository server in another zone, a "gateway" must be created. A gateway object contains two resource (agent) machines, for example, GatewayResource1 and GatewayResource2—each configured to communicate with the other. One gateway resource resides in the *source* zone and the other in the *target* zone. A gateway is bidirectional and informs the ElectricFlow server that each gateway machine is configured to communicate with its other gateway machine (in another zone).

Property Name	Description
port2	<code>number</code> : The port number used by <i>Resource2</i> - defaults to the port number used by the resource.
propertySheet	<code>reference</code> : <code>propertySheet</code>
resourceName1	<code>string</code> : The name of your choice for the first of two required gateway resources. Do not include "spaces" in a resource name.
resourceName2	<code>string</code> : The name of your choice for the second of two required gateway resources. Do not include "spaces" in a resource name.

[\[back to top\]](#)**Object Type: group**

Description: This is any *group* of users known to the ElectricFlow server, including groups defined locally within the server and those groups defined in external repositories such as LDAP or ActiveDirectory.

Property Name	Description
acl	<code>reference</code> : <code>acl</code>
createTime	<code>date</code> : The time when this object was created.
groupId	<code>id</code> : The unique ElectricFlow-generated group ID.
groupName	<code>name</code> : This is this group's name.

Object Type: group

Description: This is any *group* of users known to the ElectricFlow server, including groups defined locally within the server and those groups defined in external repositories such as LDAP or ActiveDirectory.

Property Name	Description
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
modifyTime	<code>date</code> : The time when this object was last modified.
mutable	<code>boolean</code> : If true, the member list of this group is editable within ElectricFlow via the web UI or the <code>modifyGroup</code> API.
owner	<code>name</code> : The person (user name) who created the object.
propertySheet	<code>reference</code> : <code>propertySheet</code>
providerName	<code>name</code> : The name of the <code>directory provider</code> that controls this group.

[\[back to top\]](#)

Object Type: job

Description: A *job* is an ElectricFlow structure associated with invoking a procedure. A new job is created each time a procedure begins to execute. The job keeps track of all data associated with the procedure's execution, such as the running time of each step and any errors that may occur during the step. ElectricFlow retains job information after the job completes so you can examine what occurred.

Property Name	Description
abortStatus	<code>string</code> : If set, indicates the step was aborted. Values will be either <code>ABORT</code> or <code>FORCE_ABORT</code> - indicating how the step was aborted.
abortedBy	<code>name</code> : This is the user who issued the "abort."

Object Type: `job`

Description: A *job* is an ElectricFlow structure associated with invoking a procedure. A new job is created each time a procedure begins to execute. The job keeps track of all data associated with the procedure's execution, such as the running time of each step and any errors that may occur during the step. ElectricFlow retains job information after the job completes so you can examine what occurred.

Property Name	Description
<code>acl</code>	<code>reference</code> : <code>acl</code>
<code>actualParameter</code>	<code>reference</code> : <code>propertySheet</code> An <code>actualParameter</code> is an object that provides the value for a parameter, which is passed to a procedure when it is invoked. Actual parameters can be provided for jobs and nested subprocedures within a job. Actual parameters are different from "formal parameters"- formal parameters define the parameters a procedure is expecting, and actual parameters provide values to use at run-time.
<code>callingState</code>	<code>string</code> : The full property path to the "calling state", which can appear on <code>subjobs</code> and <code>subworkflows</code> of a workflow.
<code>callingStateId</code>	<code>id</code> : The ElectricFlow-generated ID number for the calling state.
<code>combinedStatus</code>	Provides more inclusive step status output - the resulting query output may contain up to three sub-elements: <code>status message properties</code>
<code>createTime</code>	<code>date</code> : The time when this object was created.
<code>credentialName</code>	<code>name</code> : The name of the <code>credential</code> being used for impersonation when the job runs commands on a resource.
<code>deleted</code>	The object was marked for background deletion. Possible values are "0" or "1". Default is "0" (not set).
<code>directoryName</code>	<code>name</code> : The name of this job's directory within each workspace for this job.

Object Type: `job`

Description: A *job* is an ElectricFlow structure associated with invoking a procedure. A new job is created each time a procedure begins to execute. The job keeps track of all data associated with the procedure's execution, such as the running time of each step and any errors that may occur during the step. ElectricFlow retains job information after the job completes so you can examine what occurred.

Property Name	Description
<code>elapsedTime</code>	<code>number</code> : The number of milliseconds between the start and end times for the job.
<code>errorCode</code>	<code>errorCode</code> : When the outcome is <code>error</code> , this property displays the error code, identifying which error occurred.
<code>errorMessage</code>	<code>string</code> : When the outcome is <code>error</code> , this property displays the error description.
<code>external</code>	<code>boolean</code> : If "true", the job is external.
<code>finish</code>	<code>date</code> : The time this job completed.
<code>jobId</code>	<code>id</code> : This is this job's ID number, which is a UUID.
<code>jobName</code>	<code>name</code> : This is this job's name.
<code>lastModifiedBy</code>	<code>name</code> : This shows who (generally, a user name) last modified this object.
<code>launchedByUser</code>	<code>name</code> : The name of the user or project principal that explicitly launched the job. This property is blank when the job is launched by a schedule.
<code>licenseWaitTime</code>	The sum of time all job steps had to wait for a license.
<code>liveProcedure</code>	<code>name</code> : The current procedure name from which this job was created – if the procedure was renamed since the job launched, this will be the procedure's new name, and if the procedure was deleted, this will be null.

Object Type: `job`

Description: A *job* is an ElectricFlow structure associated with invoking a procedure. A new job is created each time a procedure begins to execute. The job keeps track of all data associated with the procedure's execution, such as the running time of each step and any errors that may occur during the step. ElectricFlow retains job information after the job completes so you can examine what occurred.

Property Name	Description
<code>liveSchedule</code>	<code>name</code> : The current schedule name that launched this job – if the schedule was renamed since the job was launched, this will be the schedule's new name, and if the schedule was deleted, this will be null.
<code>modifyTime</code>	<code>date</code> : The time when this object was last modified.
<code>outcome</code>	The overall result of the job: <code>success</code> , <code>warning</code> , <code>error</code> , or <code>skipped</code> .
<code>owner</code>	<code>name</code> : The person (user name) who created the object.
<code>priority</code>	Values can be <code>low</code> , <code>normal</code> (default), <code>high</code> , or <code>highest</code> .
<code>procedureName</code>	<code>name</code> : The name of the <code>procedure</code> that defines the job's steps.
<code>projectName</code>	<code>name</code> : The name of the <code>project</code> that contains this job.
<code>propertySheetId</code>	<code>reference</code> : <code>propertySheet</code>
<code>resourceWaitTime</code>	The sum of time all job steps had to wait for a resource. This could indicate that eligible resources for the step have reached their step limit, are in-use but the step requires a resource exclusively, or resources are down.
<code>runAsUser</code>	<code>name</code> : The name of the user being impersonated in this job.
<code>scheduleName</code>	<code>name</code> : The schedule name that launched this job – this field differs from <code>liveSchedule</code> in that it is written at job creation time only, and not changed, even if the schedule is renamed or deleted.
<code>start</code>	<code>date</code> : The time when this job began executing.

Object Type: `job`

Description: A *job* is an ElectricFlow structure associated with invoking a procedure. A new job is created each time a procedure begins to execute. The job keeps track of all data associated with the procedure's execution, such as the running time of each step and any errors that may occur during the step. ElectricFlow retains job information after the job completes so you can examine what occurred.

Property Name	Description
<code>status</code>	<p>Possible values are:</p> <p><code>pending</code> - the job was created, but it is waiting for prerequisite steps to complete</p> <p><code>runnable</code> - the job step is waiting for a resource</p> <p><code>scheduled</code> - the job was assigned a resource, but the command has not started running</p> <p><code>running</code> - the job/job step is running a command on the assigned resource</p> <p><code>completed</code> - the job/job step has completed</p>
<code>totalWaitTime</code>	The total sum of license, resource, a precondition, and workspace wait times job steps were restricted and had to wait to process.
<code>workspaceWaitTime</code>	The sum of time all job steps had to wait for a workspace.

[\[back to top\]](#)

Object Type: `jobStep`

Description: A *jobStep* is an instance of a step that ran - a single invocation of a command on a resource or a single call to a subprocedure within the context of a [job](#).

Property Name	Description
<code>abortStatus</code>	<code>string</code> : If set, indicates the step was aborted. Values will be either <code>ABORT</code> or <code>FORCE_ABORT</code> - indicating how the step was aborted.
<code>abortedBy</code>	<code>name</code> : The user who issued the "abort."

Object Type: `jobStep`

Description: A *jobStep* is an instance of a step that ran - a single invocation of a command on a resource or a single call to a subprocedure within the context of a [job](#).

Property Name	Description
<code>acl</code>	<code>reference</code> : acl
<code>actualParameter</code>	<code>reference</code> : propertySheet An <code>actualParameter</code> is an object that provides the value for a parameter, which is passed to a procedure when it is invoked. Actual parameters can be provided for jobs and nested subprocedures within a job. Actual parameters are different from "formal parameters"- formal parameters define the parameters a procedure is expecting, and actual parameters provide values to use at run-time.
<code>alwaysRun</code>	<code>boolean</code> : If true, this step runs even if the job is aborted before the step completes. Note that a "force abort" will abort an <code>alwaysRun</code> step.
<code>assignedResourceName</code>	<code>name</code> : The name of the resource assigned to this step by the step scheduler.
<code>broadcast</code>	<code>boolean</code> : If true, this step runs on all resources in a pool.
<code>combinedStatus</code>	Provides more inclusive step status output - the resulting query output may contain up to three sub-elements: <code>status message properties</code>
<code>command</code>	<code>string</code> : This property specifies the command this step runs.
<code>condition</code>	<code>string</code> : If this element is not present, the event is ALWAYS triggered. If specified, the event is triggered only if the value of the condition argument is TRUE; if not true, a boolean value of "false" or "0" was used. Condition arguments can be a literal, a fixed value string, or a string subject to property expansion.
<code>conditionExpanded</code>	<code>boolean</code> : The result of the expansion on the step condition.
<code>createTime</code>	<code>date</code> : The time when this object was created.

Object Type: `jobStep`

Description: A *jobStep* is an instance of a step that ran - a single invocation of a command on a resource or a single call to a subprocedure within the context of a [job](#).

Property Name	Description
<code>delayUntil</code>	For a step that was rescheduled due to a resource or workspace problem, this is the next time the step will be eligible to run.
<code>elapsedTime</code>	number : The number of milliseconds between the start and end times for the <code>jobStep</code> .
<code>errorCode</code>	errorCode : This property appears when the outcome is <code>error</code> and displays the error code, identifying which error occurred.
<code>errorHandling</code>	<p>This is the error handling policy copied from the procedure step and indicates how the step responds to errors:</p> <p><code>failProcedure</code> - The current procedure continues, but the overall status is error (default).</p> <p><code>abortProcedure</code> - Aborts the current procedure, but allows already-running steps in the current procedure to complete.</p> <p><code>abortProcedureNow</code> - Aborts the current procedure and terminates running steps in the current procedure.</p> <p><code>abortJob</code> - Aborts the entire job, terminates running steps, but allows <code>alwaysRun</code> steps to run.</p> <p><code>abortJobNow</code> - Aborts the entire job and terminates all running steps, including <code>alwaysRun</code> steps.</p> <p><code>failProcedure</code> - The current procedure continues, but the overall status is error (default).</p> <p><code>ignore</code> - Continues as if the step succeeded.</p>
<code>errorMessage</code>	string : When the outcome is <code>error</code> , this property displays an error message description.
<code>exclusive</code>	boolean : If true, this step acquires and retains its resource exclusively.

Object Type: `jobStep`

Description: A *jobStep* is an instance of a step that ran - a single invocation of a command on a resource or a single call to a subprocedure within the context of a [job](#).

Property Name	Description
<code>exclusiveMode</code>	<code>string</code> : Possible values are: none, job, step, call
<code>exitCode</code>	<code>number</code> : This is this step's exit code.
<code>external</code>	<code>boolean</code> : If "true", the job is external.
<code>finish</code>	<code>date</code> : The time when this job step completed.
<code>hostName</code>	<code>name</code> : The name of the host where this step was invoked (copied from the resource)
<code>job</code>	<code>id</code> : This is the ID number of the job that owns the job step. The string representation of the job is its name, so <code>\$(job)</code> evaluates to the job name, but <code>\$(job/foo)</code> is also legal and refers to the foo property of the job.
<code>jobId</code>	<code>id</code> : This is this job's ID number, which is a UUID.
<code>jobName</code>	<code>name</code> : This is the name of this step's job.
<code>jobStepId</code>	<code>id</code> : This is this step's ID number.
<code>lastModifiedBy</code>	<code>name</code> : This shows who (generally, a user name) last modified this object.
<code>licenseWaitTime</code>	The length of time this job step had to wait to process because the license limit was reached or exceeded.
<code>liveProcedure</code>	<code>string</code> : The current procedure name for the procedure step from which the job or job step was created – if the procedure step was renamed since the job or job step was launched, this is the procedure step's new name, and if the procedure step was deleted, this will be null.

Object Type: `jobStep`

Description: A *jobStep* is an instance of a step that ran - a single invocation of a command on a resource or a single call to a subprocedure within the context of a [job](#).

Property Name	Description
<code>liveProcedureStep</code>	name: This property shows the current procedure name for the procedure step from which this job step was created – if the procedure step was renamed since the job was launched, this will be the procedure step's new name, and if the procedure step was deleted, this will be null.
<code>logFileName</code>	name: The name of the log file produced by this step, relative to the job's workspace directory.
<code>modifyTime</code>	date: The time when this object was last modified.
<code>outcome</code>	string: The overall result of the job - possible values are: <code>success</code> , <code>warning</code> , <code>error</code> , or <code>skipped</code>
<code>owner</code>	name: The person (user name) who created the object.
<code>parallel</code>	boolean: If true, this step runs in parallel with other adjacent steps also marked to run in parallel.
<code>postExitCode</code>	number: This is this step's post processor exit code.
<code>postLogFileName</code>	name: The log file name produced by this step's post processor.
<code>postProcessor</code>	string: The post processor name used to gather information about this step.
<code>precondition</code>	string: By default, if a step has no precondition, it will run when scheduled. Set this property to make a step wait until one or more dependent conditions are met. When a job step is eligible to transition from pending to runnable, a <i>precondition</i> is evaluated. A <i>precondition</i> is a fixed text or text embedding property reference that is evaluated to TRUE or FALSE. An empty string, a <code>"0"</code> or <code>"false"</code> is interpreted as FALSE. Any other result string is interpreted as TRUE. The step will block until the precondition is TRUE.

Object Type: `jobStep`

Description: A *jobStep* is an instance of a step that ran - a single invocation of a command on a resource or a single call to a subprocedure within the context of a [job](#).

Property Name	Description
<code>procedureName</code>	<code>name</code> : The name of the procedure that contains this <code>jobStep</code> .
<code>projectName</code>	<code>name</code> : The name of the project that contains this <code>jobStep</code> .
<code>propertySheet</code>	<code>reference</code> : propertySheet
<code>releaseExclusive</code>	<code>boolean</code> : A Boolean value indicating whether this step should release its resource upon completion.
<code>releaseMode</code>	<code>string</code> : Possible values are: none, release, releaseToJob
<code>resourceName</code>	<code>name</code> : The name of the resource or pool this step should use to run on.
<code>resourceSource</code>	This property indicates whether the resource for the job step is explicitly specified by the step or was defaulted from the procedure: procedure or <code>procedureStep</code> .
<code>resourceWaitTime</code>	The length of time this job step stalled because it could not get a resource. This could indicate that eligible resources for the step have reached their step limit, are in-use but the step requires a resource exclusively, or resources are down.
<code>retries</code>	<code>number</code> : This is a number - the number of retries before the request times out.
<code>runAsUser</code>	<code>name</code> : The name of the user being impersonated in this job step.
<code>runnable</code>	<code>date</code> : The time when the step became runnable.
<code>runTime</code>	<code>number</code> : The number of milliseconds the step command spent running on a resource.

Object Type: `jobStep`

Description: A *jobStep* is an instance of a step that ran - a single invocation of a command on a resource or a single call to a subprocedure within the context of a [job](#).

Property Name	Description
<code>shell</code>	name : The shell used to execute the step's commands on a resource. The script name is inserted into the command at the position of a " {0} " marker in the command, or appended as a final argument if no marker is present. A filename suffix adjacent to the marker will be appended to the script name. For more information on shells, see the Shell definition in the Step Help topic.
<code>start</code>	date : The time when this job step began executing.
<code>status</code>	<p>Possible values are:</p> <p><code>pending</code> - the job step was created, but it is waiting for prerequisite steps to complete</p> <p><code>runnable</code> - the job step is waiting for a resource</p> <p><code>scheduled</code> - the job step was assigned a resource, but the command has not started running</p> <p><code>running</code> - the job/job step is running a command on the assigned resource</p> <p><code>completed</code> - the job/job step has completed</p>
<code>stepName</code>	name : This is this step's name.
<code>subprocedure</code>	name : The nested procedure name to call when this step executes.
<code>subproject</code>	name : This property appears if the subprocedure element is present and specifies the project to which the subprocedure belongs (by default, the current project is used.)
<code>timeLimit</code>	number : The maximum length of time this step is allowed to run.
<code>timeout</code>	date : The time when this job step will be automatically aborted if it has not yet completed.

Object Type: `jobStep`

Description: A *jobStep* is an instance of a step that ran - a single invocation of a command on a resource or a single call to a subprocedure within the context of a [job](#).

Property Name	Description
<code>totalWaitTime</code>	The sum of resource, workspace, and license wait times for this job step.
<code>waitTime</code>	number : The number of milliseconds the step spent between runnable and running (for example, waiting for a resource).
<code>workingDirectory</code>	name : The name of this step's working directory.
<code>workspaceName</code>	name : The workspace name used by the <code>jobStep</code> object.
<code>workspaceWaitTime</code>	The time this job step had to wait because no workspace was found or available.

[\[back to top\]](#)

Object Type: logEntry

Description: A *logEntry* is an object containing various types of information available in a log file generated from anywhere in the ElectricFlow system.

The *Event Log* can be configured for auto-deletion. By default, Event Log retention is 30 days. The ElectricFlow server automatically marks old log entries for deletion and the background deleter cleans out old log entries.

To change the default settings, go to Administration > Server > Settings and modify the Event log retain time and the Maximum background delete delay fields

Note: Setting the "retain" period to "0" disables the automatic deletion mechanism, allowing you to use an external cleanup script to implement a custom cleanup policy.

Property Name	Description
category	(currently not used)
container	<code>string</code> : Typically, this is the type and name of the workflow or job with a corresponding ID.
containerName	<code>name</code> : The name of the container.
containerType	<code>string</code> : The type of object the log entry pertains to (for example, procedure, job step, step).
deleted	<code>byte</code> : The object was marked for background deletion. Possible values are "0" or "1". Default is "0" (which means "deleted" is not set).
logEntryId	<code>id</code> : The log entry ElectricFlow-generated ID.
message	<code>string</code> : The message text will either be informational or display the warning or error message. The message may contain important information about a resource or workspace issue.
principal	<code>string</code> : The user or project principal from the session that was active when the event occurred.

Object Type: logEntry

Description: A *logEntry* is an object containing various types of information available in a log file generated from anywhere in the ElectricFlow system.

The *Event Log* can be configured for auto-deletion. By default, Event Log retention is 30 days. The ElectricFlow server automatically marks old log entries for deletion and the background deleter cleans out old log entries.

To change the default settings, go to Administration > Server > Settings and modify the Event log retain time and the Maximum background delete delay fields

Note: Setting the "retain" period to "0" disables the automatic deletion mechanism, allowing you to use an external cleanup script to implement a custom cleanup policy.

Property Name	Description
severity	<code>string</code> : Severity can be either TRACE, DEBUG, INFO, WARN, ERROR
subject	<code>string</code> : The object associated with the message.
subjectName	<code>name</code> : The name of the object associated with the message.
subjectType	<code>string</code> : (similar to <code>container</code>) Refers to the object the event concerns. This may be the same as the container, or it may be a different object that is related to the event in some manner.
time	<code>string</code> : The time the event was logged.

[\[back to top\]](#)

Object Type: procedure

Description: A *procedure* describes a series of actions to be performed on one or more resources. A procedure contains steps and properties and can define formal parameters.

Property Name	Description
acl	<code>reference: acl</code>
createTime	<code>date</code> : The time when this object was created.
credentialName	<code>name</code> : The name of the credential assigned to this job step.
description	<code>string</code> : A user-specified text description of the object.
jobNameTemplate	<code>name</code> : The template used to name jobs created from this procedure.
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.
procedureId	<code>id</code> : This is this procedure's ID number.
procedureName	<code>name</code> : This is this procedure's name.
projectName	<code>name</code> : The name of the <code>project</code> that contains this procedure.
propertySheet	<code>reference: propertySheet</code>
resourceName	<code>name</code> : The name of the resource or pool this procedure should use to run on.
workspaceName	<code>name</code> : The workspace name used by the procedure object.

[\[back to top\]](#)

Object Type: project

Description: A *project* is an object used in ElectricFlow to organize information. A project contains procedures, schedules, credentials, and properties.

Property Name	Description
acl	reference: acl
createTime	date : The time when this object was created.
credentialName	name : The name of the credential assigned to this project.
deleted	The object was marked for background deletion. Possible values are "0" or "1". Default is "0" (not set).
description	string : A user-specified text description of the object.
lastModifiedBy	name : This shows who (generally, a user name) last modified this object.
modifyTime	date : The time when this object was last modified.
owner	name : The person (user name) who created the object.
pluginName	name : The name of the plugin associated with this project.
projectId	id : This is this project's ID.
projectName	name : The name of the project.
propertySheet	reference: propertySheet
resourceName	name : The name of the resource.
workspaceName	name : This is the workspace name used by the project.

[\[back to top\]](#)

Object Type: property

Description: A *property* is a string value with a name. Properties can have arbitrary names and values. You can attach properties to any object in the ElectricFlow system, such as a project, procedure, or job. After a property is created, it is stored in the ElectricFlow database. You can retrieve and/or modify the value later, and you can delete properties you no longer need. Properties provide a flexible and powerful mechanism to manage data about your builds.

Note: The name "properties" is NOT a valid property name.

Property Name	Description
createTime	<code>date</code> : The time when this object was created.
description	<code>string</code> : A user-specified text description of the object.
expandable	<code>boolean</code> : If set to true, the property value will undergo string expansion when it is retrieved.
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.
canonicalPath	<code>string</code> : The path that specifies the property object.
propertyId	<code>id</code> : This property's ID number.
propertyName	<code>name</code> : This property's name.
propertySheetId	<code>reference</code> : <code>propertySheet</code> – If the property is a nested property sheet, the <code>propertySheet</code> refers to the nested sheet.
value	<code>string</code> : The property or actual parameter's value - if this property is a string property.

[\[back to top\]](#)

Object Type: `propertySheet`

Description: A property value can be a simple string or a nested *propertySheet* containing its own properties. Property sheets can be nested to any depth, which allows you to create hierarchical collections of information.

Most objects have an associated property sheet that contains "custom properties" created by user scripts.

Property Name	Description
<code>acl</code>	<code>reference</code> : <code>acl</code>
<code>createTime</code>	<code>date</code> : The time when this object was created.
<code>lastModifiedBy</code>	<code>name</code> : This shows who (generally, a user name) last modified this object.
<code>modifyTime</code>	<code>date</code> : The time when this object was last modified.
<code>owner</code>	<code>name</code> : The person (user name) who created the object.
<code>propertySheetId</code>	<code>id</code> : This property sheet's ID.

[\[back to top\]](#)

Object Type: `repository`

Description: A *repository* is an object that stores artifact versions. This object primarily contains information on how to connect to a particular artifact repository. Similar to steps in a procedure, repository objects are in a user-specified order. When retrieving artifact versions, repositories are queried in this order until one containing the desired artifact version is found.

Property Name	Description
<code>acl</code>	<code>reference</code> : <code>acl</code>
<code>createTime</code>	<code>date</code> : The time when this object was created.
<code>description</code>	<code>string</code> : A user-specified text description for this object.

Object Type: repository

Description: A *repository* is an object that stores artifact versions. This object primarily contains information on how to connect to a particular artifact repository. Similar to steps in a procedure, repository objects are in a user-specified order. When retrieving artifact versions, repositories are queried in this order until one containing the desired artifact version is found.

Property Name	Description
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.
propertySheet	<code>reference</code> : <code>propertySheet</code>
repositoryDisabled	<code>boolean</code> : Determines whether the repository is disabled. Default is "false".
repositoryId	<code>id</code> : The ElectricFlow-generated ID number of the repository.
repositoryIndex	<code>number</code> : The order of the repository, within a list of repositories.
repositoryName	<code>name</code> : The name of the repository.
url	<code>string</code> : The server URL is in the form <code>protocol://host:port/</code> . Typically, the repository server is configured to listen on port 8200 for <code>https</code> requests, so a typical URL looks like <code>https://host:8200/</code> .
zoneName	<code>name</code> : The name of the zone where this repository is or will reside.

[\[back to top\]](#)

Object Type: resource

Description: A *resource* is associated with a server machine available to ElectricFlow for running steps. A resource has a logical name in addition to a host name. A resource can be associated with one or more pools. The resource may be a "proxy target" machine that communicates with the ElectricFlow server through an ElectricFlow agent proxy machine.

Property Name	Description
acl	<code>reference: acl</code>
agentState	<code>string</code> : Specific information about an agent, including the state of the agent. Possible values are: <code>unknown alive down</code>
artifactCacheDirectory	<code>string</code> : The directory on the agent host where retrieved artifacts are stored.
createTime	<code>date</code> : The time when this object was created.
description	<code>string</code> : A user-specified text description of the object.
exclusiveJobId	<code>id</code> : The ID number of the job that owns this resource, which occurs when one of the job's steps requests exclusive use of the resource for the duration of the job.
exclusiveJobName	<code>name</code> : The name of the job that owns this resource, which occurs when one of the job's steps requests exclusive use of the resource for the duration of the job.
exclusiveJobStepId	<code>id</code> : The ID number of the job step that owns this resource, which occurs when one of the steps request exclusive use of the resource for the duration of the job.
exclusiveJobStepName	<code>name</code> : The name of the job step that owns this resource, which occurs when one of the steps request exclusive use of the resource for the duration of the job.
gateways	The gateway names associated with this resource.

Object Type: `RESOURCE`

Description: A *resource* is associated with a server machine available to ElectricFlow for running steps. A resource has a logical name in addition to a host name. A resource can be associated with one or more pools. The resource may be a "proxy target" machine that communicates with the ElectricFlow server through an ElectricFlow agent proxy machine.

Property Name	Description
<code>hostName</code>	<code>name</code> : The computer name or IP address for the machine containing the ElectricFlow agent for this resource.
<code>hostOS</code>	<code>string</code> : The full name of the host operating system, plus its version. However, if this host is a proxy, "proxied" appears as the host description/name.
<code>hostPlatform</code>	<code>string</code> : Examples for "platform" are: Windows, Linux, HPUX, and so on. However, if this host is a proxy, "proxied" appears as the <code>hostPlatform</code> name.
<code>lastModifiedBy</code>	<code>name</code> : This shows who (generally, a user name) last modified this object.
<code>lastRunTime</code>	<code>date</code> : The most recent time a job step ran on the resource.
<code>modifyTime</code>	<code>date</code> : The time when this object was last modified.
<code>owner</code>	<code>name</code> : The person (user name) who created the object.
<code>pools</code>	<code>string</code> : A space-separated list of one or more pool names where this resource is a member. Steps defined to run on a resource pool will run on any available member (resource) in the pool.
<code>port</code>	<code>number</code> : The port number for this resource.
<code>propertySheet</code>	<code>reference</code> : <code>propertySheet</code>
<code>proxyCustomization</code>	<code>string</code> : This property displays the customization of the ElectricFlow proxy agent.

Object Type: resource

Description: A *resource* is associated with a server machine available to ElectricFlow for running steps. A resource has a logical name in addition to a host name. A resource can be associated with one or more pools. The resource may be a "proxy target" machine that communicates with the ElectricFlow server through an ElectricFlow agent proxy machine.

Property Name	Description
proxyHostName	<code>name</code> : The name of the ElectricFlow agent being used to proxy to another agent, the proxy target.
proxyPort	<code>number</code> : The port number of the ElectricFlow agent being used to proxy to another agent, the proxy target.
proxyProtocol	<code>name</code> : The name of the proxy agent protocol used for communication from the ElectricFlow proxy agent to the proxy target - default is SSH.
repositoryNames	A "new line" separated list of repository names.
resourceDisabled	<code>boolean</code> : A Boolean value indicating whether this resource was disabled.
resourceId	<code>id</code> : This is this resource's ID.
resourceName	<code>name</code> : The name of the resource or pool.
shell	<code>name</code> : The shell used to execute the step's commands on a resource. If no shell was defined on a step, the shell defined on the resource is used, or if no shell was defined on the resource, a default shell is used. For shells: The script name is inserted into the command at the position of a "{0}" marker in the command, or appended as a final argument if no marker is present. A filename suffix adjacent to the marker will be appended to the script name. For more information on shells, see the Shell definition in the Step Help topic.
stepCount	<code>number</code> : The current number of executing steps on this resource.
stepLimit	<code>number</code> : This property specifies the maximum number of steps that can run on this resource at one time.

Object Type: resource

Description: A *resource* is associated with a server machine available to ElectricFlow for running steps. A resource has a logical name in addition to a host name. A resource can be associated with one or more pools. The resource may be a "proxy target" machine that communicates with the ElectricFlow server through an ElectricFlow agent proxy machine.

Property Name	Description
trusted	<code>boolean</code> : If "true", this agent is trusted.
useSSL	<code>boolean</code> : A Boolean value indicating whether this resource uses SSL for communication. Transport Layer Security (TLS) has replaced Secure Sockets Layer version 3.0 (SSLv3) on the ElectricFlow web server and the ElectricFlow server.
workspaceName	<code>name</code> : The workspace name used by this resource.
zoneName	<code>string</code> : The name of the zone where this resource resides.

[\[back to top\]](#)

Object Type: resourcePool

Description: A *resource pool* is a container for a group of resources.

Property Name	Description
acl	<code>reference</code> : <code>acl</code>
autoDelete	<code>boolean</code> : If "true", the resource pool is deleted when the last resource is removed or deleted.
createTime	<code>date</code> : The time when this object was created.
description	<code>string</code> : A user-specified text description of the object.

Object Type: resourcePool

Description: A *resource pool* is a container for a group of resources.

Property Name	Description
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
lastResourceUsed	<code>name</code> : The name of the most recently used resource from the pool.
modifyTime	<code>date</code> : The time when this object was last modified.
orderingFilter	<code>string</code> : A Javascript block invoked when scheduling resources for a pool. Note: A Javascript block is not required unless you need to override the default resource ordering behavior.
owner	<code>name</code> : The person (user name) who created the object.
propertySheet	<code>reference</code> : propertySheet
resourcePoolDisabled	<code>boolean</code> : A Boolean value indicating whether this resource pool was disabled.
resourcePoolId	<code>id</code> : This resource pool's ID number.
resourcePoolName	<code>name</code> : The name of the resource pool.

[\[back to top\]](#)

Object Type: resourceUsage

Description: Provides usage information for all resources in the system.

For any job step running on a resource, there is a resource usage record that contains the ID and name of the job, job step, and resource.

Property Name	Description
jobId	<code>id</code> : This is this job's ID number, which is a UUID.

Object Type: resourceUsage

Description: Provides usage information for all resources in the system.

For any job step running on a resource, there is a resource usage record that contains the ID and name of the job, job step, and resource.

Property Name	Description
jobName	<code>name</code> : The name of the job.
jobStepId	<code>id</code> : The ID number of the job step.
jobStepName	<code>name</code> : The name of the job step.
licenseWaitTime	The time this job step had to wait because no license was found or available.
resourceId	<code>id</code> : The ElectricFlow-generated resource ID number.
resourceName	<code>name</code> : The name (or names) of the resource.
resourcePoolId	<code>id</code> : The ElectricFlow-generated resource pool's ID number.
resourcePoolName	<code>name</code> : The name of the resource pool.
resourceUsageId	<code>id</code> : The unique ID of the resource usage record.
resourceWaitTime	The time this job step had to wait because no resource was found or available. This could indicate that eligible resources for the step have reached their step limit, are in-use but the step requires a resource exclusively, or resources are down.
waitReason	Possible values are: <code>license</code> , <code>resource</code> , or <code>workspace</code> .
workspaceWaitTime	The time this job step had to wait because no workspace was found or available.

[\[back to top\]](#)

Object Type: <code>schedule</code> Description: <i>Schedules</i> are used to execute procedures and determine when specific procedures run.	
Property Name	Description
<code>acl</code>	<code>reference</code> : <code>acl</code>
<code>actualParameter</code>	<code>reference</code> : <code>propertySheet</code> An <i>actualParameter</i> is an object that provides the value for a parameter, which is passed to a procedure when it is invoked. Actual parameters can be provided for jobs and nested subprocedures within a job. Actual parameters are different from "formal parameters"- formal parameters define the parameters a procedure is expecting, and actual parameters provide values to use at run-time.
<code>beginDate</code>	<code>string</code> : The first day on which the schedule is active, in the form YYYY-MM-dd. For example, "2009-06-25"
<code>createTime</code>	<code>date</code> : The time when this object was created.
<code>credentialName</code>	<code>name</code> : The name of the <code>credential</code> being used for impersonation when the schedule launches a job.
<code>description</code>	<code>string</code> : A user-specified text description of the object.
<code>endDate</code>	<code>string</code> : The end of the range of dates the schedule is active, in the form: YYYY-MM-dd. The actual end date is not included in the range. For example, "2009-06-25"
<code>interval</code>	<code>string</code> : A floating point number that represents the time to wait between invocations of the schedule.
<code>intervalUnits</code>	<code>string</code> : These are the units to use when interpreting the <i>interval</i> value. The <i>units</i> can be <code>hours</code> , <code>minutes</code> , <code>seconds</code> , or <code>continuous</code> .
<code>lastModifiedBy</code>	<code>name</code> : This shows who (generally, a user name) last modified this object.
<code>lastRunTime</code>	<code>date</code> : The last time a job was launched by a schedule.

Object Type: <code>schedule</code> Description: <i>Schedules</i> are used to execute procedures and determine when specific procedures run.	
Property Name	Description
<code>misfirePolicy</code>	string: The policy the schedule uses when it is unable to launch a job at the scheduled time: <code>ignore</code> - wait until the next scheduled time to run <code>runOnce</code> - run immediately, then resume normal scheduling
<code>modifyTime</code>	date: The time when this object was last modified.
<code>monthDays</code>	string: This property specifies which days of the month this schedule should run - shown as a space-separated list of numbers (1-31).
<code>owner</code>	name: The person (user name) who created the object.
<code>priority</code>	string: Values can be <code>low</code> , <code>normal</code> (default), <code>high</code> , or <code>highest</code>
<code>procedureName</code>	name: The name of the procedure that contains this schedule.
<code>projectName</code>	name: The name of the project that contains this schedule.
<code>propertySheet</code>	reference: propertySheet
<code>scheduleDisabled</code>	boolean: A Boolean value indicating whether this schedule was disabled.
<code>scheduleId</code>	id: This schedule's ID number.
<code>scheduleName</code>	name: This property displays this schedule's name and differs from the <code>liveSchedule</code> property found under a job in that the <code>liveSchedule</code> property is written at job creation time only, and not changed, even if the schedule is renamed or deleted.
<code>startTime</code>	string: The time of day when this schedule should start running, in the form: <code>HH:mm:ss</code>

Object Type: schedule

Description: *Schedules* are used to execute procedures and determine when specific procedures run.

Property Name	Description
stopTime	string: The time of day when this schedule should stop running, in the form: HH:mm:ss
timeZone	string: A Java-compatible time zone string.
weekDays	string: This property specifies which days of the week this schedule should run. This is a space-separated list of MONDAY TUESDAY WEDNESDAY THURSDAY FRIDAY SATURDAY SUNDAY. (Note: These are not localized strings.)

[\[back to top\]](#)

Object Type: server

Description: This is the ElectricFlow server. There is only one such object in your database.

Read-only server properties

Two read-only server properties are available for use in a step, a script, or an email notifier, which can be used to identify the ElectricFlow server location.

`/server/hostname` - returns the ElectricFlow server name

`/server/hostIP` - returns the ElectricFlow server IP address

These properties are especially useful for building a URL link to an ElectricFlow web page. For example, a link in an email notifier that links back to a Job Details page.

A sample link: `https://$[/server/hostname]/commander/jobDetails.php?jobId=$[jobId]`

Property Name	Description
acl	reference: <code>acl</code>
createTime	date: The time when this object was created.
hostIP	string: The server's IP address.

Object Type: `server`

Description: This is the ElectricFlow server. There is only one such object in your database.

Read-only server properties

Two read-only server properties are available for use in a step, a script, or an email notifier, which can be used to identify the ElectricFlow server location.

`/server/hostname` - returns the ElectricFlow server name

`/server/hostIP` - returns the ElectricFlow server IP address

These properties are especially useful for building a URL link to an ElectricFlow web page. For example, a link in an email notifier that links back to a Job Details page.

A sample link: `https://$[/server/hostname]/commander/jobDetails.php?jobId=$[jobId]`

Property Name	Description
<code>hostname</code>	<code>name</code> : Generally refers to the computer name or IP address for the machine containing the ElectricFlow server.
<code>lastModifiedBy</code>	<code>name</code> : This shows who (generally, a user name) last modified this object.
<code>modifyTime</code>	<code>date</code> : The time when this object was last modified.
<code>owner</code>	<code>name</code> : The person (user name) who created the object.
<code>propertySheet</code>	<code>reference</code> : <code>propertySheet</code>
<code>serverId</code>	<code>id</code> : The server's ID number.
<code>serverName</code>	<code>string</code> : The ElectricFlow server's name - defaults to "server".

[\[back to top\]](#)

Object Type: state

Description: A *state* object belongs to a workflow and corresponds to the state definition, including a pointer to the workflow, formal parameters (cloned from definition), expanded actual parameters (from the last time a transition was taken to this state), and the "process" which includes the procedure or workflow (cloned from definition), unexpanded actual parameters (cloned from definition, expanded and passed to the process on entry), and the "last run" entity reference.

Property Name	Description
acl	<code>reference</code> : <code>acl</code>
active	<code>boolean</code> : If "true", the state of the workflow is active.
actualParameter	<code>reference</code> : <code>propertySheet</code> An <code>actualParameter</code> is an object that provides the value for a parameter, which is passed to a procedure when it is invoked. Actual parameters can be provided for jobs and nested subprocedures within a job. Actual parameters are different from "formal parameters"- formal parameters define the parameters a procedure is expecting, and actual parameters provide values to use at run-time.
createTime	<code>date</code> : The time when this object was created.
description	<code>string</code> : A user-specified text description of the object.
errorMessage	<code>string</code> : When the outcome is <code>error</code> , this property displays an error message description.
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.
projectName	<code>name</code> : The name of the <code>project</code> that contains this state.
propertySheet	<code>reference</code> : <code>propertySheet</code>
stateId	<code>id</code> : The ElectricFlow-generated ID number for the state.

Object Type: state

Description: A *state* object belongs to a workflow and corresponds to the state definition, including a pointer to the workflow, formal parameters (cloned from definition), expanded actual parameters (from the last time a transition was taken to this state), and the "process" which includes the procedure or workflow (cloned from definition), unexpanded actual parameters (cloned from definition, expanded and passed to the process on entry), and the "last run" entity reference.

Property Name	Description
stateName	name : The name of the state.
subjob	name : The name of the subjob.
subparameters	reference : propertySheet
subprocedure	name : The name of the nested procedure called when a step runs.
subproject	string : If a subprocedure argument was used, this is the name of the project where that subprocedure is found. By default, the current project is used.
substartingState	name : Name of the starting state for the workflow launched when the state is entered.
subworkflow	name : The name of the subworkflow - a workflow called by another workflow.
subworkflowDefinition	name : The name of the subworkflow definition.
workflow	name : The name of the workflow.

[\[back to top\]](#)

Object Type: stateDefinition

Description: A *stateDefinition* is a named object that belongs to a workflow definition, which includes specifications for whether or not the state is "startable", formal parameters, notifications, and the process. A process includes a procedure or workflow, the starting state (for workflows only), and actual parameters.

Property Name	Description
acl	<code>reference: acl</code>
actualParameter	<code>reference: propertySheet</code> An <i>actualParameter</i> is an object that provides the value for a parameter, which is passed to a procedure when it is invoked. Actual parameters can be provided for jobs and nested subprocedures within a job. Actual parameters are different from "formal parameters"- formal parameters define the parameters a procedure is expecting, and actual parameters provide values to use at run-time.
createTime	<code>date</code> : The time when this object was created.
description	<code>string</code> : A user-specified text description of the object.
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.
projectName	<code>name</code> : The name of the <code>project</code> that contains this state definition.
propertySheet	<code>reference: propertySheet</code>
startable	<code>boolean</code> : "True" means this state definition can be the initial state of an instantiated workflow.
stateDefinitionId	<code>id</code> : The ElectricFlow-generated ID number for the state definition.
stateDefinitionName	<code>name</code> : The name of the state definition.

Object Type: stateDefinition

Description: A *stateDefinition* is a named object that belongs to a workflow definition, which includes specifications for whether or not the state is "startable", formal parameters, notifications, and the process. A process includes a procedure or workflow, the starting state (for workflows only), and actual parameters.

Property Name	Description
subprocedure	<code>name</code> : The name of the nested procedure called when a step runs.
subproject	<code>string</code> : If a subprocedure argument was used, this is the name of the project where that subprocedure is found. By default, the current project is used.
substartingState	<code>name</code> : Name of the starting state for the workflow launched when the state is entered.
subworkflowDefinition	<code>name</code> : The name of the subworkflow definition.
workflowDefinitionName	<code>name</code> : The name of the workflow definition.

[\[back to top\]](#)

Object Type: step

Description: A *step* includes a command or script executed on a single resource and is the smallest unit of work ElectricFlow understands. Each step specifies a resource on which it should run (either the name of a specific machine or the name of a *pool* of equivalent machines, in which case ElectricFlow picks a machine from the pool). A step can be given a time limit and if the step does not complete within the specified time, ElectricFlow automatically aborts it.

Steps are ordered within a procedure and normally execute sequentially. However, it is possible to mark a consecutive range of steps for parallel execution, so all steps in that range run concurrently.

A step in one procedure can invoke another procedure instead of running a command. The invoking step provides parameters needed by the nested procedure.

Property Name	Description
<code>acl</code>	<code>reference</code> : <code>acl</code>
<code>actualParameter</code>	<code>reference</code> : <code>propertySheet</code> An <code>actualParameter</code> is an object that provides the value for a parameter, which is passed to a procedure when it is invoked. Actual parameters can be provided for jobs and nested subprocedures within a job. Actual parameters are different from "formal parameters"- formal parameters define the parameters a procedure is expecting, and actual parameters provide values to use at run-time.
<code>alwaysRun</code>	<code>boolean</code> : If true, this step runs even if the job is aborted before the step completes. Note that a force abort will abort an <code>alwaysRun</code> step.
<code>broadcast</code>	<code>boolean</code> : If true, this step will run on all resources in a pool.
<code>command</code>	<code>string</code> : This property specifies the command this step runs.
<code>condition</code>	<code>string</code> : If this element is not present, the event is ALWAYS triggered. If specified, the event is triggered only if the value of the condition argument is TRUE; if not true, a boolean value of "false" or "0" was used. Condition arguments can be a literal, a fixed value string, or a string subject to property expansion.
<code>createTime</code>	<code>date</code> : The time when this object was created.
<code>credentialName</code>	<code>name</code> : The credential name assigned to this step.

Object Type: step

Description: A *step* includes a command or script executed on a single resource and is the smallest unit of work ElectricFlow understands. Each step specifies a resource on which it should run (either the name of a specific machine or the name of a *pool* of equivalent machines, in which case ElectricFlow picks a machine from the pool). A step can be given a time limit and if the step does not complete within the specified time, ElectricFlow automatically aborts it.

Steps are ordered within a procedure and normally execute sequentially. However, it is possible to mark a consecutive range of steps for parallel execution, so all steps in that range run concurrently.

A step in one procedure can invoke another procedure instead of running a command. The invoking step provides parameters needed by the nested procedure.

Property Name	Description
description	<code>string</code> : A user-specified text description of the object.
errorHandling	<p>This is the error handling policy copied from the procedure step and indicates how the step responds to errors:</p> <p><code>failProcedure</code> - The current procedure continues, but the overall status is error (default).</p> <p><code>abortProcedure</code> - Aborts the current procedure, but allows already-running steps in the current procedure to complete.</p> <p><code>abortProcedureNow</code> - Aborts the current procedure and terminates running steps in the current procedure.</p> <p><code>abortJob</code> - Aborts the entire job, terminates running steps, but allows <code>alwaysRun</code> steps to run.</p> <p><code>abortJobNow</code> - Aborts the entire job and terminates all running steps, including <code>alwaysRun</code> steps.</p> <p><code>ignore</code> - Continues as if the step succeeded.</p>
exclusive	<code>boolean</code> : If true, this step acquires and retains its resource exclusively.
exclusiveMode	<code>string</code> : Possible values are: none, job, step, call
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.

Object Type: step

Description: A *step* includes a command or script executed on a single resource and is the smallest unit of work ElectricFlow understands. Each step specifies a resource on which it should run (either the name of a specific machine or the name of a *pool* of equivalent machines, in which case ElectricFlow picks a machine from the pool). A step can be given a time limit and if the step does not complete within the specified time, ElectricFlow automatically aborts it.

Steps are ordered within a procedure and normally execute sequentially. However, it is possible to mark a consecutive range of steps for parallel execution, so all steps in that range run concurrently.

A step in one procedure can invoke another procedure instead of running a command. The invoking step provides parameters needed by the nested procedure.

Property Name	Description
logFileName	<code>name</code> : The name of the log file produced by this step, relative to the job's workspace directory.
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.
parallel	<code>boolean</code> : If true, this step runs in parallel with other adjacent steps also marked to run in parallel.
postLogFileName	<code>name</code> : This property displays the log file name produced by this step's post processor.
postProcessor	<code>string</code> : This property displays the post processor name that is used to gather information about this step. Typically, this is either <code>postp</code> or <code>postp <options></code> , or an appropriate command or path including options to a postprocessor available on the ElectricFlow server.

Object Type: step

Description: A *step* includes a command or script executed on a single resource and is the smallest unit of work ElectricFlow understands. Each step specifies a resource on which it should run (either the name of a specific machine or the name of a *pool* of equivalent machines, in which case ElectricFlow picks a machine from the pool). A step can be given a time limit and if the step does not complete within the specified time, ElectricFlow automatically aborts it.

Steps are ordered within a procedure and normally execute sequentially. However, it is possible to mark a consecutive range of steps for parallel execution, so all steps in that range run concurrently.

A step in one procedure can invoke another procedure instead of running a command. The invoking step provides parameters needed by the nested procedure.

Property Name	Description
precondition	<p>string: By default, if a step has no precondition, it will run when scheduled. Set this property to make a step wait until one or more dependent conditions are met. When a job step is eligible to transition from pending to runnable, a <i>precondition</i> is evaluated.</p> <p>A <i>precondition</i> is a fixed text or text embedding property reference that is evaluated to TRUE or FALSE. An empty string, a \"0\" or \"false\" is interpreted as FALSE. Any other result string is interpreted as TRUE. The step will block until the precondition is TRUE.</p>
procedureName	name: The name of the procedure that contains this <i>step</i> .
projectName	name: The name of the project that contains this step.
propertySheet	reference: propertySheet
releaseExclusive	boolean: A Boolean value indicating whether this step should release its resource upon completion.
releaseMode	string: Possible values are: none, release, releaseToJob
resourceName	name: The resource's name this step should use to run on.

Object Type: step

Description: A *step* includes a command or script executed on a single resource and is the smallest unit of work ElectricFlow understands. Each step specifies a resource on which it should run (either the name of a specific machine or the name of a *pool* of equivalent machines, in which case ElectricFlow picks a machine from the pool). A step can be given a time limit and if the step does not complete within the specified time, ElectricFlow automatically aborts it.

Steps are ordered within a procedure and normally execute sequentially. However, it is possible to mark a consecutive range of steps for parallel execution, so all steps in that range run concurrently.

A step in one procedure can invoke another procedure instead of running a command. The invoking step provides parameters needed by the nested procedure.

Property Name	Description
shell	<code>name</code> : The shell used to execute the step's commands on a resource. The script name is inserted into the command at the position of a "{0}" marker in the command, or appended as a final argument if no marker is present. A filename suffix adjacent to the marker will be appended to the script name. For more information on shells, see the Shell definition in the Step Help topic.
stepId	<code>id</code> : The step's ID number.
stepName	<code>name</code> : The step's name.
subprocedure	<code>name</code> : The nested procedure name to call when this step executes.
subproject	<code>name</code> : This property is displayed if the subprocedure element is present and specifies the project to which the subprocedure belongs (by default, the current project is used.)
timeLimit	<code>number</code> : A floating point number that specifies the maximum length of time this step is allowed to run.
timeLimitUnits	The units to use when interpreting the time limit. Units can be <code>hours</code> , <code>minutes</code> , or <code>seconds</code> .
workingDirectory	<code>name</code> : The name of the step's working directory.
workspaceName	<code>name</code> : The workspace name used by this step.

[\[back to top\]](#)

Object Type: transition

Description: A *transition* object belongs to a state and corresponds to the transition definition, and includes the target state (unexpanded actual parameters cloned from definition, expanded and passed to the target state on entry), the Javascript condition, and the trigger (onEnter|onStart|onCompletion|manual).

Property Name	Description
acl	<code>reference: acl</code>
actualParameter	<code>reference: propertySheet</code> An <i>actualParameter</i> is an object that provides the value for a parameter that is passed to the target state when the transition is taken.
condition	<code>string</code> : If empty or non-zero, the transition is allowed to trigger. If set to "0", the transition is ignored.
createTime	<code>date</code> : The time when this object was created.
description	<code>string</code> : A user-specified text description of the object.
index	The numeric index of a transition that indicates the transition order in the state definition's transition list.
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.
projectName	<code>name</code> : The name of the <code>project</code> that contains this transition.
propertySheet	<code>reference: propertySheet</code>
stateName	<code>name</code> : The name of the state.
targetState	<code>string</code> : The target state for the transition definition.
transitionId	<code>id</code> : The ElectricFlow-generated ID for the transition.

Object Type: transition

Description: A *transition* object belongs to a state and corresponds to the transition definition, and includes the target state (unexpanded actual parameters cloned from definition, expanded and passed to the target state on entry), the Javascript condition, and the trigger (onEnter|onStart|onCompletion|manual).

Property Name	Description
transitionName	<code>name</code> : The name of the transition.
trigger	<code>string</code> : Possible values are: onEnter - before any actions onStart - after a subjob or workflow is created onCompletion - after a subjob or workflow completes manual - when a user manually requests a transition
workflowName	<code>name</code> : The name of the workflow.

[\[back to top\]](#)**Object Type: transitionDefinition**

Description: A *transitionDefinition* is a named object that belongs to a state definition, which includes the target state definition (with actual parameters to the target state, the Javascript condition, and the trigger (onEnter|onStart|onCompletion|manual).

Property Name	Description
acl	<code>reference</code> : <code>acl</code>
actualParameter	<code>reference</code> : <code>propertySheet</code> An <i>actualParameter</i> is an object that provides the value for a parameter passed to the target state.
condition	<code>string</code> : If empty or non-zero, the transition is allowed to trigger. If set to "0", the transition is ignored.
createTime	<code>date</code> : The time when this object was created.

Object Type: transitionDefinition

Description: A *transitionDefinition* is a named object that belongs to a state definition, which includes the target state definition (with actual parameters to the target state, the Javascript condition, and the trigger (onEnter|onStart|onCompletion|manual)).

Property Name	Description
description	<code>string</code> : A user-specified text description of the object.
index	The numeric index of a transition that indicates the transition order in the state definition's transition list.
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.
projectName	<code>name</code> : The name of the project that contains this transition definition.
propertySheet	<code>reference</code> : propertySheet
stateDefinitionName	<code>name</code> : The name of the state definition.
targetState	<code>string</code> : The target state for the transition definition.
transitionDefinitionId	<code>id</code> : The ElectricFlow-generated ID for the transition definition.
transitionDefinitionName	<code>name</code> : The name of the transition definition.
trigger	<code>string</code> : Possible values are: onEnter – before any actions onStart – after a subjob or workflow is created onCompletion – after a subjob or workflow completes manual – when a user manually requests a transition
workflowDefinitionName	<code>name</code> : The name of the workflow definition.

[\[back to top\]](#)

Object Type: `User`

Description: *User* refers to any user currently logged into the ElectricFlow system or any user who may use ElectricFlow, but may not be currently logged in.

Property Name	Description
<code>acl</code>	<code>reference:</code> <code>acl</code>
<code>createTime</code>	<code>date:</code> The time when this object was created.
<code>email</code>	<code>name:</code> Displays this user's email address.
<code>fullUserName</code>	<code>name:</code> The user's real name.
<code>lastModifiedBy</code>	<code>name:</code> This shows who (generally, a user name) last modified this object.
<code>modifyTime</code>	<code>date:</code> The time when this object was last modified.
<code>mutable</code>	<code>boolean:</code> If true, the user is editable within ElectricFlow via the web UI or the <code>modifyUser</code> API.
<code>owner</code>	<code>name:</code> The person (user name) who created the object.
<code>propertySheet</code>	<code>reference:</code> <code>propertySheet</code>
<code>providerName</code>	<code>name:</code> The name of the <code>directory provider</code> that controls this user.
<code>userId</code>	<code>id:</code> A unique ID number for a user object.
<code>userName</code>	<code>name:</code> Displays this user's name and also appears in a group object and displays the name of a user who belongs to this group.

[\[back to top\]](#)

Object Type: workflow

Description: A *workflow* object includes a pointer to the workflow definition, sets of states, the active state, the starting state, parameters to the initial state, "complete" - a boolean value determining whether or not the workflow is complete, and a log containing the transactions taken and user events.

Property Name	Description
acl	<code>reference</code> : acl
activeState	<code>string</code> : The name of the active state on the workflow object.
actualParameter	<code>reference</code> : <code>propertySheet</code> An <i>actualParameter</i> is an object that provides the value for a parameter that is passed to the target state when the transition is taken.
callingState	<code>string</code> : The full property path to the state that created this workflow.
callingStateId	<code>id</code> : The ID number for the full property path to the state that created this workflow.
completed	<code>boolean</code> : If "true", the workflow is completed and no additional transactions will be evaluated.
createTime	<code>date</code> : The time when this object was created.
deleted	<code>boolean</code> : The object was marked for background deletion. Possible values are "0" or "1". Default is "0" (not set).
elapsedTime	The time this object ran from start to finish.
finish	<code>date</code> : The date/time when this object finished.
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
launchedByUser	<code>string</code> : The name of the user or project principal that explicitly launched the workflow.
liveWorkflowDefinition	<code>name</code> : The current workflow definition name for the workflow definition from which the workflow was created.

Object Type: workflow

Description: A *workflow* object includes a pointer to the workflow definition, sets of states, the active state, the starting state, parameters to the initial state, "complete" - a boolean value determining whether or not the workflow is complete, and a log containing the transactions taken and user events.

Property Name	Description
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.
projectName	<code>name</code> : The name of the project that contains this workflow.
propertySheet	<code>reference</code> : propertySheet
start	<code>date</code> : The date/time when this workflow began to run.
startingState	<code>string</code> : The initial state of the workflow.
workflowDefinitionName	<code>name</code> : The name of the workflow definition.
workflowId	<code>id</code> : The ElectricFlow-generated ID for the workflow.
workflowName	<code>name</code> : The name of the workflow.

[\[back to top\]](#)**Object Type: workflowDefinition**

Description: A *workflowDefinition* object contains state and transition definitions, including the workflow name template (analogous to a job name template on a procedure), and ordered sets of state and transition definitions.

Property Name	Description
acl	<code>reference</code> : acl
createTime	<code>date</code> : The time when this object was created.

Object Type: workflowDefinition

Description: A *workflowDefinition* object contains state and transition definitions, including the workflow name template (analogous to a job name template on a procedure), and ordered sets of state and transition definitions.

Property Name	Description
description	<code>string</code> : A user-specified text description of the object.
lastModifiedBy	<code>name</code> : This shows who (generally, a user name) last modified this object.
modifyTime	<code>date</code> : The time when this object was last modified.
owner	<code>name</code> : The person (user name) who created the object.
projectName	<code>name</code> : The name of the <code>project</code> that contains this workflow definition.
propertySheet	<code>reference</code> : <code>propertySheet</code>
workflowDefinitionId	<code>id</code> : The ElectricFlow-generated ID for the workflow definition.
workflowDefinitionName	<code>name</code> : The name of the workflow definition.
workflowNameTemplate	<code>string</code> : Template used to determine the default names for workflows launched from a workflow definition.

[\[back to top\]](#)

Object Type: workspace

Description: ElectricFlow provides each job step with an area on the disk it can use for "working files" and results. This disk area is called a job *workspace*. (The job workspace defaults to a directory under the workspace directory.) A job step can create whatever files it needs within its workspace, and ElectricFlow automatically places files such as step logs in the workspace.

Normally, a single workspace is shared by all steps in a job, but different steps within a job could use different workspaces. The location of the job step workspace is displayed on the Job Details page for the job under "Details" for the step.

Property Name	Description
acl	<code>reference:</code> acl
agentDrivePath	<code>name:</code> The drive-letter based mount point for this workspace on Windows agents.
agentUncPath	<code>name:</code> The UNC path for this workspace on Windows agents.
agentUnixPath	<code>name:</code> The UNIX path for this workspace on UNIX agents.
createTime	<code>date:</code> The time when this object was created.
credentialName	<code>name:</code> The name of the credential being used when a resource connects to a network share while setting up the workspace.
description	<code>string:</code> A user-specified text description of the object.
lastModifiedBy	<code>name:</code> This shows who (generally, a user name) last modified this object.
local	<code>boolean:</code> If "true", this workspace is local.
modifyTime	<code>date:</code> The time when this object was last modified.
owner	<code>name:</code> The person (user name) who created the object.
propertySheet	<code>reference:</code> propertySheet

Object Type: workspace

Description: ElectricFlow provides each job step with an area on the disk it can use for "working files" and results. This disk area is called a job *workspace*. (The job workspace defaults to a directory under the workspace directory.) A job step can create whatever files it needs within its workspace, and ElectricFlow automatically places files such as step logs in the workspace.

Normally, a single workspace is shared by all steps in a job, but different steps within a job could use different workspaces. The location of the job step workspace is displayed on the Job Details page for the job under "Details" for the step.

Property Name	Description
<code>workspaceDisabled</code>	<code>boolean</code> : A Boolean value that indicates whether this workspace was disabled.
<code>workspaceId</code>	<code>id</code> : The workspace's ID number.
<code>workspaceName</code>	<code>name</code> : The workspace's name.
<code>zoneName</code>	<code>name</code> : The name of the zone where this workspace resides.

[\[back to top\]](#)

Object Type: Zone

Description: ElectricFlow provides the ability to create zones—often used to enhance security.

- A zone is a way to partition a collection of agents to secure them from use by other groups. For example, you might choose to create a developers zone, a production zone, and a test zone—agents in one zone cannot directly communicate with agents in another zone.
- A *default* zone is created during ElectricFlow installation. The server implicitly belongs to the default zone, which means all agents in this zone can communicate with the server directly (without the use of a gateway).
- Each zone can have one or more "gateway agents", which you define. Gateway agents are used for communication from one zone to another zone. For more information, see the [Gateways](#) Help topic.

Property Name	Description
<code>acl</code>	<code>reference</code> : <code>acl</code>

Object Type: Zone

Description: ElectricFlow provides the ability to create zones—often used to enhance security.

- A zone is a way to partition a collection of agents to secure them from use by other groups. For example, you might choose to create a developers zone, a production zone, and a test zone—agents in one zone cannot directly communicate with agents in another zone.
- A *default* zone is created during ElectricFlow installation. The server implicitly belongs to the default zone, which means all agents in this zone can communicate with the server directly (without the use of a gateway).
- Each zone can have one or more "gateway agents", which you define. Gateway agents are used for communication from one zone to another zone. For more information, see the [Gateways](#) Help topic.

Property Name	Description
createTime	date : The time when this object was created.
description	string : A user-specified text description for the object.
lastModifiedBy	name : This person (generally, a user name) last modified this object.
modifyTime	date : The time when this object was last modified.
owner	name : The person (user name) who created the object.
propertySheet	reference : propertySheet
resources	string : A space-separated list of resources contained in this zone.
zoneId	id : The ElectricFlow-generated ID for this zone.
zoneName	string : The name of the zone.

Property error codes

The following list provides error codes that may appear as a value within an `errorCode` property.

ABORTED

ACCESS_DENIED

AGENT_BAD_WORKINGDIR

AGENT_FAILED_CONNECT_BROKER

AGENT_FAILED_CREATE_FILE
AGENT_FAILED_CREATE_WORKSPACE
AGENT_FAILED_IMPERSONATION
AGENT_FAILED_MAP_DRIVE
AGENT_FAILED_PROXYTARGET_PING
AGENT_INCOMPATIBLE_VERSION
AGENT_INTERNAL_ERROR
AGENT_INVALID_CWD
AGENT_INVALID_MESSAGE
AGENT_INVALID_PING_TOKEN
AGENT_INVALID_WORKSPACE
AGENT_IO_CONNECTION_REFUSED
AGENT_IO_CONNECTION_RESET
AGENT_IO_ERROR
AGENT_IO_NO_ROUTE_TO_HOST
AGENT_IO_PORT_UNREACHABLE
AGENT_MALFORMED_XML
AGENT_NONABSOLUTE_PATH
AGENT_NONEXISTENT_DIR
AGENT_NONEXISTENT_FILE
AGENT_PING_FAILED
AGENT_STREAM_STOPPED
AGENT_TIMEOUT
AGENT_UNKNOWN_CMDID
AGENT_UNKNOWN_COMMAND
AGENT_WRONG_FILE_TYPE
BAD_AGENT_RESPONSE
BAD_LOGFILE_PATH
CANCELED
CIRCULAR_PROCEDURE_REFERENCE
CORRUPT_CREDENTIAL

DUPLICATE_JOB_NAME
EMPTY_SUBPROCEDURE
FAILED_JOB_RENAME
FORMAL_PARAMETER_ERROR
INTERNAL_ERROR
INVALID_EMAIL_HEADER
MY_EVENT_EXPANSION_ERROR
NO_EMAIL_HEADERS_SEPARATOR
NONEXISTENT_EMAIL_CONFIG
NONEXISTENT_PROCEDURE
NONEXISTENT_RESOURCE
NONEXISTENT_WORKSPACE
NOTIFIER_EXPANSION_ERROR
POST_PROCESSOR_ERROR
POST_PROCESSOR_OUTPUT_FAILURE
PROPERTY_REFERENCE_ERROR
RESOURCE_WITHOUT_HOSTNAME
SERVER_SHUTDOWN
TIMEOUT
UNKNOWN_HOST

Postprocessors: Collecting Data for Reports

[Postp](#)

[Extending postp: matchers](#)

[Postp functions](#)

[Integration with the ElectricFlow user interface](#)

[Postp integration with Java Tools](#)

Overview

In ElectricFlow, reporting is divided into two phases.

- The first phase is data collection: interesting information is extracted from job step logs and saved in the ElectricFlow database.
- The second phase is report generation: data collected previously is retrieved and organized into reports.

The report phases are separated so data is gathered once but then available to use in a variety of different reports either immediately or later. For example, one report might summarize errors within a particular job, and another report might display error trends from all jobs over the past month.

ElectricFlow implements data collection with a *postprocessor*.

- A postprocessor is a command associated with a particular procedure step. If a postprocessor is specified for a step, it executes concurrently with the main step command.
- The postprocessor runs on the same machine as the main command and in the same working directory, and it retrieves the log file from the step as its standard input.

ElectricFlow includes a standard postprocessor called *postp* that you can use and extend.

postp scans the step's log file looking for interesting output such as error messages and then sets properties on the job step to describe what it found. For example, *postp* might create a property named "errors" whose value is a count of the number of error messages in the log file, or a property named "tests" that counts the number of tests executed by the step. Also, *postp* can extract portions of the step log that contain useful diagnostic information and save this information for reporting.

Standard ElectricFlow reports, such as those on the Job Details page, display information collected by *postp* such as properties named "errors" and diagnostic log extracts. This information is available immediately, even before the step completes execution, so you can view it via ElectricFlow's web interface to monitor step execution. Also, you can create additional report generators of your own, which can use the same information displayed by ElectricFlow and/or any other additional information of your choice.

Postp

ElectricFlow's general-purpose postprocessor, *postp*, uses regular expression patterns to detect interesting lines in a step log.

- *postp* is already configured with patterns to handle many common cases such as error messages and warnings from *gcc*, *gmake*, *cl*, *junit*, and *cppunit*, or any error message containing the string "error." You can use the empty matcher group named "none" if just want to run a *postpEndHook*.
- *postp* is easy to use by simply setting the postprocessor for a step to "postp."
- *postp* also supports several useful command-line options.

To explore these options, invoke "*postp --help*" from your command-line.

Extending *postp*: matchers

If you find useful patterns in your log files undetected by *postp*, you can extend *postp* with additional patterns. This feature is easily implemented, but the extension interface currently involves writing simple Perl scripts and you need to look at the Perl source file for *postp* as you do this. The *postp* source code is installed during ElectricFlow installation and located in the `src/postp.pl` file in the distribution directory.

Postp is driven by a collection of *matchers*, which are regular expression patterns that select certain lines from step logs, and by a collection of Perl functions the matchers invoke to handle lines of interest. A matcher is a Perl hash with three values similar to the following example:

```
{
  id      => "error",
  pattern => q{ERROR:|[Ee]rror:},
  action  => q{
    incValue("errors"); diagnostic("", "error", -4)
  },
}
```

Explanation of the values in the Perl hash example above:

- **id** - A unique name for the matcher—used to identify the matcher in command-line arguments and a few other places.
- **pattern** - A regular expression tested against each line of the step log file. This particular pattern matches lines containing any of the strings "ERROR:", "Error:" or "error:"
- **action** - A Perl script that executes whenever a log line matches the pattern for this matcher. The script in this example increments a variable named "errors" that is copied automatically to a job step property with the same name. The script also saves a portion of the step log beginning 4 lines prior and extending through the current line and associates those lines with this error so it can be displayed in the web interface. See below for more information on the `incValue` and `diagnostic` functions.

To extend postp with patterns of your own, write a Perl script to add new patterns to the `@::gMatchers` array. Here is a simple example:

```
my @newMatchers = (
  {
    id      => "coreDump",
    pattern => "core dumped",
    action  => q{
      incValue("coreDumps")
    },
  },
  {
    id      => "segFault",
    pattern => "segmentation fault",
    action  => q{
      incValue("segFaults")
    },
  },
);
push @::gMatchers, @newMatchers;
```

These matchers detect lines containing the strings "core dumped" or "segmentation fault" and increment separate variables for each line type.

After writing extension code, you must ask postp to execute the code when it starts up. You can execute extension code in one of two ways:

- Place the code into a file and invoke postp with the `--load` option.
For example, `postp --load fileName`
- Copy the code into a property in ElectricFlow and use the `--loadProperty` option to postp.
For example, `postp --loadProperty /myProject/extraMatchers`

Postp extensions can contain arbitrary Perl code, which means you can use this mechanism to define additional functions to invoke in matcher actions if existing functions do not provide what you need.

Note: Additional postp matchers are available for your convenience. The matcher sample directory was installed during ElectricFlow installation. Go to `src/samples/postp`.

Postp functions

Postp contains several built-in functions to invoke in your matchers. The most useful functions are summarized below. Also, you can scan postp code for additional functions.

debugLog(format, arg, arg, ...)

Outputs information to the debug log, if the `--debugLog` command-line switch is set. Format provides a format string similar to `printf`, and each argument provides a value to substitute into the format string.

backTo(pattern, start)

This function searches backward to find the first line in the step log matching "pattern" (a regular expression) and returns the offset of that line relative to the current line. The result is normally used as an argument to the "diagnostic" function. "Start" is optional; it specifies the first line to check and is specified as an offset relative to the current line (it defaults to -1).

backWhile(pattern, start)

This function searches backward to find the first line in the step log that does not match "pattern" (a regular expression) and returns the offset relative to the current line of the line just after the first non-matching line. The result is normally used as an argument to the "diagnostic" function. "Start" is optional; it specifies the first line to check and is specified as an offset relative to the current line (it defaults to -1).

currentModule()

Returns the name of the current module (as determined by previous calls to `pushModule` and `popModule`), or an empty string if there is no current module.

diagnostic(name, type, first, last)

This function extracts a group of contiguous lines from the log file and saves them along with additional information for reports such as the log extracts, which appear at the bottom of the Job Details web page. The range of lines to extract is indicated by "first" and "last," each of which is an offset relative to the current line. For example, if "first" is -3 and "last" is 2, then 6 lines will be recorded: 3 lines before the current line, the current line, and 2 lines after the current line. In many cases, the values for "first" and "last" are computed by calling functions such as "forwardTo" or "backWhile." "Name" provides an identifier for this particular diagnostic, such as the name of a test that failed or a file that did not compile. "Type" specifies which kind of information this is, and must be "error," "warning," or "info." In addition to log lines, this function records "name," "type," the current module, if any, and the name of the current matcher.

forwardTo(pattern, start)

This function searches forward to find the first line in the step log matching "pattern" (a regular expression) and returns the offset of that line relative to the current line. The result is normally used as an argument to the "diagnostic" function. "Start" is optional; it specifies the first line to check and is specified as an offset relative to the current line (it defaults to 1).

forwardWhile(pattern, start)

This function searches forward to find the first line in the step log that does not match "pattern" (a regular expression) and returns the offset relative to the current line of the line just before the first non-matching line. The result is normally used as an argument to the "diagnostic" function. "Start" is optional; it specifies the first line to check and is specified as an offset relative to the current line (it defaults to 1).

incValue(name, increment)

Adds "increment" to a value named "name" and arranges for that value to be written eventually to a property by the same name on the current job step. If this is the first call for "name," its value is initialized to 0.

"Increment" is optional and defaults to 1.

Note: postp does not check the job step for a pre-existing property with the same name; it simply overwrites it.

logLine(lineNumber)

Returns the line from the step log given by lineNumber. 1 corresponds to the first line in the step log and the Perl variable `$_:gCurrentLine` holds the number of the current line. This function caches a sliding window of lines in the file, allowing you to go back to retrieve lines preceding the current line (as long as they do not precede it by too many lines). If the requested line is off the end of the file then `undef` is returned. If the requested line is before the beginning window of cached lines, an empty string is returned.

popModule()

Cancels the effect of the most recent call to `pushModule`, resetting the current module name to whatever it was before the corresponding call to `pushModule`.

postpEndHook()

If you define a function with this name, it invokes after `postp` has finished processing the log file, but before it makes its final properties update on the job step. Use this function to perform your own operations such as generating an error if the log file did not contain a particular line you were expecting.

pushModule(name)

In some situations it is possible to divide the log file into parts corresponding to different modules. For example, with recursive make invocations, there are typically notifications in the log output before and after each recursive make. This function is invoked to indicate a new module is being entered, where "name" is the name of the module. After this function is called, the "diagnostic" function will include "name" with error or warning messages to provide additional information in job reports. The previous module name, if any, is saved; you can return to it by calling `popModule`.

setProperty

This function sets a property in the ElectricFlow server. If the named parameter is a relative path, like `moduleCount`, the property is set or created on the current job step. You can use an absolute path, like `/myJob/fileLocation` also. Calling `setProperty` does not result in an immediate call to the ElectricFlow server. The property is added to an update list for updating at the next "update interval", typically every 30 seconds.

Integration with the ElectricFlow user interface

`postp` interacts with the ElectricFlowUI using two methods. The first method: Create custom properties with special names that are recognized by the UI itself. The second method: Create a file that contains "diagnostics",

which are used to display errors or warnings, and link them to specific sections in the step's log file.

Custom property names and values

`postp` can be used to create properties in ElectricFlow, on the job step or anywhere else in ElectricFlow. However, the following properties are used by the standard ElectricFlow UI, so you should use these property names whenever possible, and avoid using these names in ways that conflict with the definitions below.

- **compiles** - the number of files compiled during the job step
- **diagFile** - the filename in the top-level directory of the job's workspace, containing diagnostics extracted from the step's log file
- **errors** - the number of errors (compilation failures, test failures, application crashes, and so on) that occurred during the job step. When property errors are set by `postp`, the step outcome is set to error also.
- **tests** - the number of tests executed by the job step, including successes and failures
- **testsSkipped** - the number of tests skipped during the job step
- **warnings** - the number of warnings that occurred during the job step. When property warnings is set by `postp`, the step outcome is set to warning also.
- **preSummary** - if this property exists, its value is displayed in the "Status" field (on the Job Details page) for this step. This property appears *before* whatever would normally be displayed for status. If the property contains multiple lines separated by newline characters, each line is displayed on a separate line in the status field.
- **postSummary** - if this property exists, its value is displayed in the "Status" field (on the Job Details page) for this step. This property appears *after* whatever would normally be displayed for status. If the property contains multiple lines separated by newline characters, each line is displayed on a separate line in the status field.
- **summary** - if this property exists, its value is displayed in the "Status" field (in the job reports) for this step, replacing whatever would normally be displayed for status. If the property contains multiple lines separated by newline characters, each line is displayed on a separate line in the status field.

Diagnostic information

The second form of postprocessor generated output contains diagnostic extracts from the step's log file, typically providing additional information about problems. The postprocessor stores diagnostic information in an XML file in the top-level directory of the job workspace, then sets the step's **diagFile** property with the name of the file.

Diagnostic files must have a format like the following example:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<diagnostics>
  <diagnostic>
    <matcher>compileError</matcher>
    <name>testa.c</name>
    <type>error</type>
    <module>util/timeLib</module>
    <firstLine>2</firstLine>
    <numLines>7</numLines>
    <message>testa.c: In function 'procl':
      testa.c:12: error: parse error before ';' token
      testa.c:13: error: 'for' loop initial declaration used outside C99 mode
      testa.c:14: error: parse error before ';' token
```

```
        testa.c:16: error: too few arguments to function 'exit'
        testa.c:18:2: warning: no newline at end of file
        testa.c:18: error: parse error at end of input
    </message>
</diagnostic>
<diagnostic>
    <name>testa.o</name>
    <type>error</type>
    <module>util/timeLib</module>
    <firstLine>9</firstLine>
    <numLines>1</numLines>
    <message>make: *** [testa.o] Error 1 </message>
</diagnostic>
</diagnostics>
```

XML elements in the diagnostic file are as follows:

- `diagnostics` - Overall container; its children consist of all of the diagnostic elements.
- `diagnostic` - Describes one diagnostic extract; the elements described below are all children of this element.
- `matcher` - (optional) Identifier for the matcher that triggered this diagnostic; used primarily for debugging.
- `name` - (optional) Identifier that indicates the problem or situation that resulted in this diagnostic, such as the name of a failed test or the name of a file that did not compile.
- `type` - Type of message: must be "error," "warning," or "info."
- `module` - (optional) Name of the module in which the issue occurred, such as the name of a source code module being compiled at the time of a compile error.
- `firstLine` - Line number in the log file for the first line of the diagnostic extract (1 means the first line of the file). This element is used to provide a link from the diagnostic extract to the full log file.
- `numLines` - Total number of lines included in the diagnostic extract.
- `message` - The actual lines from the log file.

Postp integration with Java Tools

ElectricFlow integrates with three standard Java tools through postp matchers. These Java tools are:

- EMMA - an open source toolkit for measuring and reporting Java code coverage (Emma v2.0)
- JUnit - a framework for writing and running automated tests (tested with Ant v1.7)
- Clover - Atlassian's Java code coverage (generated by system, functional, or unit tests) analysis tool (Clover v2.4)

Note: If one of these Java tools is invoked in an ElectricFlow job step, ElectricFlow automatically detects the invocation [if you are using Postp] and adds any reports generated by these tools to the list of links at the top of the Job Details page.

The process...

Postp parses output from the invoked JAVA tool to match paths to reports it has already created. Then, postp copies all files that comprise the report to a unique location in the "Artifacts" Directory. Next, postp generates a

link to the location in the Artifacts Directory to make the report available in the Links section on the Job Details page.

The following example of generated output from Emma illustrates this process:

```
emma:

init:
[mkdir] Created dir: C:\Documents and Settings\ptharani\Desktop\emma\emma-2.0.5312\examples\out

compile:
[javac] Compiling 4 source files to C:\Documents and Settings\ptharani\Desktop\emma\emma-2.0.5312\examples\out

run:
[emmajava] EMMA: processing classpath ...
[emmajava] EMMA: [3 class(es) processed in 47 ms]
[emmajava] main(): running doSearch()...
[emmajava] main(): done
[emmajava] EMMA: writing [txt] report to [C:\Documents and Settings\ptharani\Desktop\emma\emma-2.0.5312\examples\coverage\coverage.txt] ...
[emmajava] EMMA: writing [html] report to [C:\Documents and Settings\ptharani\Desktop\emma\emma-2.0.5312\examples\coverage\coverage.html] ...

all:

BUILD SUCCESSFUL
Total time: 1 second
```

From this output...

The actual link may reside here:

```
/home/commanderWorkspace/job_112_
200901081732/artifacts/javaTools/238/emmaCoverage/2/coverage.html
```

While the link name may be:

```
Step Id 238 ant-on-the-fly - emma report# 2
```

And the value of the link may be:

```
jobSteps/238//javaTools/238/emmaCoverage/2/coverage.html
```

Java Tool matcher examples

Two examples of postp Emma matchers:

```
{
  id      => "emmaReport1",
  pattern => q{EMMA: writing},
  action  => q{emmaExtractReport()},
},
{
  id      => "emmaReport2",
  pattern => q{\\[report\\] writing},
  action  => q{emmaValidateOutput()},
},
```

An example of postp JUnit matchers:

```
{
  id      => "junitReportCapture",
  pattern => q{\\[junitreport\\] Processing},
  action  => q{junitExtractReport ()},
},
```

These matchers correspond to the following output:

```
junit.report:
[junitreport] Processing
/net/WinStor2home/ptharani/junitDemo1/sample/testreport/TESTS-TestSuites.xml to
/tmp/null11214791178
[junitreport] Loading stylesheet jar:[file:/usr/local/tools/common/apache-ant-
1.7.0/lib/ant-junit.jar!/org/apache/tools/ant/taskdefs/optional/junit/xsl/junit-
frames.xsl
[junitreport] [file:/usr/local/tools/common/apache-ant-1.7.0/lib/ant-
junit.jar!/org/apache/tools/ant/taskdefs/optional/junit/xsl/junit-frames.xsl
[junitreport]] Transform time: 622ms
[junitreport] Deleting: /tmp/null11214791178
```

An example of postp Clover matchers:

```
{
  id      => "cloverHtmlReportAntTask",
  pattern => q{\\[clover-html-report\\] Writing HTML report to},
  action  => q{cloverExtractReport ()},
},
```

These matchers correspond to the following output:

```
clover.report:
[clover-html-report] Clover Version 2.4.0, built on November 05 2008 (build-747)
[clover-html-report] Loaded from: /home/cloverDemo/clover-ant-2.4.0/lib/clover.jar
[clover-html-report] Clover: Evaluation License registered to electric cloud.
[clover-html-report] You have 27 day(s) before your license expires.
[clover-html-report] Loading coverage database from: '/home/cloverDemo/clover-ant-
2.4.0/tutorial/.clover/clover2_4_0.db'
[clover-html-report] Writing HTML report to '/home/cloverDemo/clover-ant-
2.4.0/tutorial/clover_html'
[clover-html-report] Done. Processed 1 packages in 2559ms (2559ms per package).
```

Artifacts directory

The value of the artifacts directory determines the scope of what is visible in the ElectricFlow UI from a job's workspace. By default, the Artifacts Directory is set to "artifacts", so the artifacts directory would have the form:

```
<path to job workspace>/artifacts
```

The value of the artifacts directory can be set in any of the following properties:

```
/myJob/artifactsDirectory
/myProject/artifactsDirectory
/server/settings/artifactsDirectory
```

Postp queries these properties [in the order listed] to determine the location of the Artifacts Directory. If no value is found (because the property was never set), the default "artifacts" is used.

Postp has a feature where it recognizes that data belongs to some standard tool (such as Junit), copies the logs produced by that tool to the artifacts directory, and then creates the report-url properties. These actions are done with the `junitReportCapture` matcher.

If you want to disable this matcher in your *postp* invocation in the step that runs junit tests, do the following:

```
postp --dontCheck junitReportCapture
```

It is possible that the artifacts directory will still be created even if nothing is put in it. If that occurs, set the `artifactsDirectory` property on your job (or the owning project) to empty-string.

Reports

ElectricFlow provides multiple reports and custom report capabilities to help you manage your build environment.

- Real-time reports - filtered view of your workload in real-time
- Build reports - summary reports produced at the end of a build and attached to the job
- Batch reports - summaries of your build environment with trends over time, two types:
 - Default Batch reports - automatically installed during ElectricFlow installation and scheduled to run daily
(Cross Project Summary, Variant Trend, Daily Summary, Resource Summary, Resource Detail)
 - Optional Batch reports - you can configure, rename, and schedule these reports to fit your requirements
(Category Report, Procedure Usage Report, Count Over Time Report, Multiple Series Reports)
- Custom reports - your choice to create and add at any time

You can also Stasd and Graphite to generate custom reports.

Note: Batch and Custom reports must be run on the ElectricFlow server's agent (local agent) only. These reports need the BIRT report engine, which is installed on the ElectricFlow server.

Run reports on a non-local resource

If you want to configure reports to run on a non-local resource, do the following:

1. In the web interface, go to Projects > Electric Cloud > runReports > runEcrtpdata.
2. Change the Parameter Resource from the default value "local" to the name of a different resource. the resource must have access to the plugins directory.

Real-time reports

Home page

The Home page is a user-customizable dashboard that can be configured to show you just the ElectricFlow jobs you care about.

The Jobs Quick View section (on the right-side of the Home page) shows jobs you define using filters on job properties. You can create multiple categories filtered on specific project names, procedure names, users, or any other job property. Summary details for the job "pop-up" if you hover your mouse over one of the jobs.

Jobs

Unlike the Home page, which allows you to see specific subsets of work, the Jobs page shows all jobs that are running and all jobs that have completed. This is a good report to view when you need to see a list of all jobs run by any user or schedule. System administrators and managers can use this page to monitor overall system throughput.

Job details / Step details

After a job starts, you can get execution details from the Job Details page. To view job details, click on the job from the Home page or the Jobs tab. From within the Job Details page you can drill down to see details for an individual step. This is the finest grain of reporting, showing the detailed results of a specific job.

Resources

The Resources page reports the current state of your defined resources. This page also shows the current state of the agent (connected or not) and any steps currently running on that resource. Use this report to monitor your resources in near real time.

Build reports using *ecreport*

ElectricFlow includes a utility, *ecreport*, that summarizes build results and sends that summary through email. This customizable utility allows you to create a simple matrix based on results stored in properties and log files. Details such as SCM change logs and errors can be included.

In addition, you can attach your own reports to any job and have them show up as attachments to the Job Details Report. To attach custom reports, put them in the top-level workspace directory in a file that ends with ".html" and contains the string "report" or "Report." Names of all matching files are displayed in the Summary section in the upper right corner of the page. Click on a report name to view the report.

The following is an example of *ecreport* output:

thunder-main.4315-200706261344 is good!

[View Online](#)

	Windows	Linux
Compile	1183 compiles	1121 compiles
Build installer	ok	ok
Agent unit tests	748 tests	642 tests
Server unit tests	1875 tests	1860 tests
Web unit tests	1556 tests	1557 tests
Tool unit tests	410 tests	338 tests
Install	ok	ok
System tests	22 tests	22 tests

General Information

Start Time	6/26/2007 13:44:19
Elapsed Time	1 hour, 27 minutes, 6.0 seconds
Workspace	//f2/scratch/buildserver3build/thunder-main.4315-200706261344

Recent Updates

----- chris -----

Change 24248 by chris@chris-thunder on 2007/06/26 13:24:31

Adding "index-redirect.php", which will be used to redirect web browsers to the "thunder" sub-directory when users only type in the host name for the web server. At install time, this file should be moved to the apache/htdocs directory, and renamed "index.php".

This will resolve the issue of https redirection not working when using the httpd.conf "Redirect" directive.

Jobs fixed ...

NMB-2761 on 2007/06/26 by tom *closed*

<https://buildserver2:443> redirects to <http://www.thunder.com/>

Affected files ...

.. //depot/thunder/main/web/docroot/index-redirect.php#1 add

ecreport extracts information about a particular job from the ElectricFlow server and writes an HTML report to standard output. The report is organized as a 2-dimensional table summarizing the results of steps in the job. The table layout can be customized by defining its format using the ectool (or ec-perl) --load option. Invoke the program in the top-level directory of the job's workspace so it can access log file extracts in that workspace (or, use the --dir option).

The following options are available for use with the *ecreport* utility:

Option	Description
--culprit	If specified and the job failed, the report will be emailed to each user mentioned in the updates file. The value of this option provides the subject line for the message (%s will be replaced with the job name).
--data	Name of file containing job data for the report. If specified, this data is used in place of querying the ElectricFlow server. Used primarily for testing.
--dir	Change to this working directory before doing anything else.
--help	Print this message and exit without doing anything.

Option	Description
<code>--jobId</code>	Identifier for the ElectricFlow job to report on (required unless <code>--data</code> is supplied). This is a UUID.
<code>--load</code>	Name of a file containing Perl code to evaluate after option processing. Used primarily for debugging. There can be multiple <code>--load</code> options.
<code>--emailConfig</code>	The name of the Email Configuration to use when emailing the report. Defaults to "default".
<code>--mailto</code>	Email the report to this addresses.
<code>--replyto</code>	Value for the "Reply-To" field in emailed reports.
<code>--server</code>	Location of the ElectricFlow server (hostname or hostname:port), to retrieve job data. Defaults to the <code>COMMANDER_SERVER</code> environment variable.
<code>--subject</code>	Value for the "Subject" field in emailed reports. If the value contains a <code>%s</code> , a string summarizing the job replaces the <code>%s</code> .
<code>--updates</code>	Name of a file containing information about recent updates reflected in the job. If specified, the contents of this file are included in the report.
<code>--version</code>	Print ecreport version number.
<code>--webRoot</code>	Base URL for the ElectricFlow Web server where the job was run, such as <code>http://myServer</code> . URLs in the report will refer to ElectricFlow Web pages relative to this URL.

Example 1

```
ecreport --jobId $[/myjob/jobId] --load ./rptformat.pl --updates update.log --
webRoot
http://myServer/commander > ecreport.html
```

This example writes a report to the file `ecreport.html`. Because this file has an `.html` extension, it will be displayed in the Reports section at the top of the Job Details page. The format of the table for this report is specified in `rptformat.pl`. An example of the contents from this file would be:

```
@::gTableRows = (
    [ "",
      "Windows",
      "Linux",
      "Solaris"],
    ["Extract",
```

```

        "span",
        "span",
        "update.log"],
["Compile",
    "windows-compile.log",
    "linux-compile.log",
    "solaris-compile.log"],
["Unit test",
    "windows-unittest.log",
    "linux-unittest.log",
    "solaris-unittest.log"],
["System test",
    "windows-systemtest.log",
    "linux-systemtest.log",
    "solaris-systemtest.log"]
);

```

The “Recent Updates” report section is specified in `update.log`. The `--webRoot` argument defines the root for any URL that appears in the report.

Example 2

```

ecreport --jobId $[/myJob/jobName] --load ./ecreportConfig.pl --mailto
user1@company.com --replyto user2@company.com
--subject 'Build Report' --updates updates.log
--webRoot http://myServer/commander

```

This report will be sent as the body of an email message to `user1@company.com` with subject “Build Report”.

Default Batch Reports (Run Reports)

These reports are installed during the ElectricFlow installation and automatically scheduled to run daily.

To access these reports: Select the Projects tab, then select the Electric Cloud project, then click the Reports subtab. This report collection graphs trends over time, including success/failure, build times, and resource consumption. These reports allow you to customize two levels of grouping into “variants.” These variants may represent products, branches, locations, architectures, or any other descriptors you would like to use to group your workload.

Note: After a Default Batch Report runs, a link for each report appears on the Job Details page in the Links section. When you select a report from the Links section, it will be displayed in full-screen view.

Report types

Cross Project Summary - This report shows the status of multiple variants over the past 30 days. This high-level summary of each of the 30 days shows green, yellow, or red depending on the result of the best build of the day. The summary also shows average times, total invocations, and other data.

Variant Trend - This report shows more details about a particular variant such as build outcome over time, build quantity over time, elapsed build times over time, and a list of actual jobs that ran during the period.

Daily Summary - This report shows the most recent results of each variant, including the last successful build.

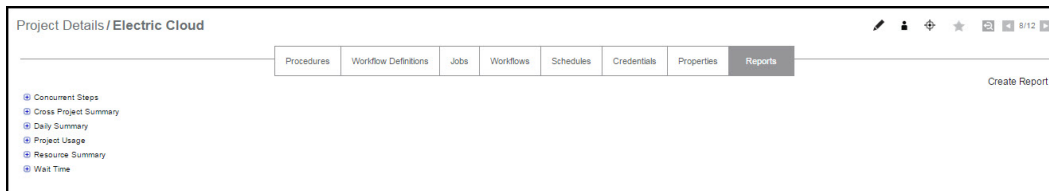
Resource Summary - This report shows resource utilization over the last 24 hours. You can see which resources are idle and which resources are busy. ElectricFlow has advanced features that allow you to select resources for your job at run time, using late binding of property names. This feature provides unparalleled flexibility and control over your environment, but could make it difficult to find out which resources were used. This report provides the visibility you need into resource loading.

Resource Detail - This report shows the utilization of a particular resource over the course of one day.

Wait Time - This report shows cumulative wait time due to license, resource, and workspace waits. It also shows the number of steps that were delayed for any of those reasons.

Default Batch Reports

- Cross Project Summary
 - Variant Trend
- Daily Summary
- Resource Summary
 - Resource Detail



The navigation tree on the left-side of the screen allows you choose the report and date of interest. These reports, which show long term trends over all projects and resources, are run from the `runReports` procedure in the Electric Cloud project.

The following example shows the Default Batch Reports, the historical reports link, and Optional Batch Reports (described below). Notice that you can expand a selected report to three levels. The third level represents drill-down links available within that report. In addition, if you click on a report name, the most current report is displayed.

**Tip**

You may change the number of reports displayed under each report type. Locate the `propertySheet` named `ec_reportConfiguration` on the project or server (project overrides server) and find the "groupings" section. Next, you can create nested property sheets under `ec_reportConfiguration` with the properties `groupingName`, `sortDirection`, and `limit`. If you set this information using `ectool`, you will also see it on ElectricFlow's web interface .

If the `ec_reportConfiguration` `propertySheet` does not exist, you will need to create it, then copy the example below into the `propertySheet`.

Example:

```
ectool setProperty ec_reportConfiguration/reportDaysLimit --value 90 --projectName foo
```

```
ectool setProperty ec_reportConfiguration/groupings/1/groupingName --value reportType
--projectName foo
ectool setProperty ec_reportConfiguration/groupings/1/sortDirection --value ascending
--projectName foo
ectool setProperty ec_reportConfiguration/groupings/1/limit --value 100 --projectName
foo

ectool setProperty ec_reportConfiguration/groupings/2/groupingName --value reportDate
--projectName foo
ectool setProperty ec_reportConfiguration/groupings/2/sortDirection --value descending
--projectName foo
ectool setProperty ec_reportConfiguration/groupings/2/limit --value 5 --projectName
foo

ectool setProperty ec_reportConfiguration/groupings/3/groupingName --value
reportResource --projectName foo
ectool setProperty ec_reportConfiguration/groupings/3/sortDirection --value ascending
--projectName foo
ectool setProperty ec_reportConfiguration/groupings/3/limit --value 100 --projectName
foo
```

This example explicitly defines the default behavior for project foo's reports.

The 1, 2 and 3 `propertySheets` define the order of the groupings. The actual names 1, 2, 3, could be anything as long as they are listed alphanumerically in the order you need.

Additionally, any groupings that do not exist will be skipped during output. For example, if you added a `reportRunByUser` grouping #4 and added it to some reports that define `reportType="User Builds"`, `reportDate`, and `reportRunByUser` properties (but not `reportResource`), you would get something similar to a resource usage report structure with users rather than resources at the deepest level.

Note: If you choose to display 500 or more reports at one time, you may experience some performance degradation.

Advanced reporting information - `ecrptdata`

How are default batch reports generated?

Reports are scheduled as part of a special procedure, `runReports`, in the Electric Cloud project. This procedure has one step, `runEcrptdata`, that runs the `ecrptdata` program. This program reads data from ElectricFlow, summarizes it for reports, and generates HTML reports using the BIRT reporting engine. The program takes input parameters. During installation, reporting is configured as follows:

```
ecrptdata --end yesterday --grain 300 --debug 2 --reports all
```

By default, all reports are run every day at 2am. The Cross Project Summary and Variant Trend shows data from the previous 30 days. The Daily Summary, Resource Summary, and Resource Detail reports show data from the last full day (yesterday). You can adjust the schedule to run more often or at different times by altering the "Report Schedule" schedule in the Electric Cloud project.

Changing the report time period

If you would like to alter the report time period, you must change the input parameters to the `ecrptdata` program within the procedure step "Electric Cloud:runReports:runEcrptdata". Select the `Ecrptdata` step name to go to the Edit Step page. Make the changes you need in the Parameters section. If the `--end` option

is omitted, the end of the reporting period is the time when the report runs (now). If the keyword "yesterday" is provided, reports will include full data for yesterday. Otherwise, a specific date and time can be given in the format "YYYY-MM-DDTHH:MM:SS".

Changing the report grouping

You can tell ElectricFlow how to group the Cross Project Summary and Variant Trend reports. By default, the outer grouping is "Project Name" and the inner grouping is "Procedure Name." To change these groupings, use the following `ecrptdata` options:

<code>--group1Name xxxx</code>	Set the column header for the first column of the Cross Project Summary report to "xxxx".
<code>--group1Property yyyy</code>	Use property "yyyy" as the name to group on in the first column of the Cross Project Summary report.

All lookups are done with respect to an individual job and will be converted to absolute property paths. The values `projectName`, `procedureName`, and `jobId` are set in context for the job being examined.

Examples:

`/myProject/foo` will be converted into the API call `"ectool getProperty /projects/projectName/foo"`

`/myProcedure/foo` will be converted into the API call `"ectool getProperty /projects/projectName/procedures/procedureName/foo"`

`/myJob/foo` will be converted into the API call `>"ectool getProperty /jobs/jobId/foo"`

`/myParameter/foo` will be converted into the API call `"ectool getActualParameter foo -jobId jobId"`

To show the value of the procedure name as the outer grouping and the value of the parameter "branch" as the inner grouping, use the following `ecrptdata` options:

```
--group1Name Procedure --group1Property "/myProcedure/procedureName"
--group2Name Branch --group2Property /myParameter/branch
```

Note: The `/myParameter` "shortcut" to the property path was created specifically for `ecrptdata` only—it cannot be used as other `/my` property shortcuts are used. For more information on property shortcuts, see the [Properties](#) Help topic. In addition, when working with `ecrptdata`, any `/my` value is case-insensitive. For example, you can use `/myParameter` or `/myparaxeter`.

Filtering

`ecrptdata` supports "findObjects" type filters when searching for jobs and job steps.

This feature is implemented with two new command line switches: `"--jobFilter <property, operator, operand>"` and `"--stepFilter <property, operator, operand>"`. The `jobFilter` option filters the CrossProjectSummary, DailySummary, and VariantTrend reports. The `stepFilter` option filters the ResourceSummary and ResourceDetail reports.

Examples:

```
--jobFilter "branch,isNotNull,1"
--jobFilter "jobName,like,my-procedure%"
--stepFilter "outcome,notEquals,aborted"
```

Adding additional data columns to the report

You can add custom data columns to each report. These columns are specified by providing a comma separated list of property references in the following ecrptdata options:

<code>--crossProjectProperties</code>	sets extra columns for the Cross Project Summary Report
<code>--variantTrendProperties</code>	sets extra columns for the Variant Trend Report
<code>--dailySummaryProperties</code>	sets extra columns for the Daily Summary Report
<code>--resourceSummaryProperties</code>	sets extra columns for the Resource Summary Report
<code>--resourceDetailProperties</code>	sets extra columns for the Resource Detail Report

You must choose properties appropriate for the type of report and grouping you have chosen. For example, if you choose Project Name and Procedure Name for your groupings in the Cross Project Summary report, each line of the report will be in the context of a project or procedure. You can add extra columns to show properties on the project or procedure, but it would not make sense to show properties on a resource, job, or job step. For resource reports, the properties must be in the context of a resource.

The column header is the name of the property.

Examples:

```
--crossProjectProperties "/myproject/QA_State,/myprocedure/PartNumber"
--resourceSummaryProperties "/myresource/OS,/myresource/Architecture"
```

Secure login

ecrptdata provides options for secure login to the ElectricFlow server.

Currently, there are three ways to login:

1. Use both the "--user <user>" and "--pass <password>" options.
This method is not secure... the password will be in plain text and visible to others.

2. If you omit the "--pass <password>" option, the password will be read from `stdin`. You can then read the password from a secure file or from a stored credential.

For example:

```
cat mySafePasswordFile.txt | ecrptdata --user <user> .....
ectool getFullCredential myCred --value password | ecrptdata --user <user> .....
```

3. Another option, "--credential <credentialName>", allows you to specify a credential name. If you use this option, "--user <user>" and "--pass <password>" are ignored. Note that the credential must be attached to the `runReports` job step. Both the user and password are retrieved from the credential.

Logic:

1. If --credential is used, attempt to lookup credential and extract user and password
2. If user is not set, return error
3. If password is not set, read from `stdin`
4. Attempt to login to server specified in --server <server> option using user and password

Optional Batch Reports

Additional reports are available, but these reports are not scheduled to run by default. You can schedule these reports to suit your purpose.

Note: When you run an Optional Batch Report, a link for each report appears on the Job Details page in the Links section. When you select a report from the Links section, it is displayed in full-screen view. The following optional reports are available:

- Category Report
- Procedure Usage Report
- Count Over Time Report
- Multiple Series Report
(formerly known as the "Value Over Time" report)

These reports are available from the Reports tab on the Project Details page for your Project.

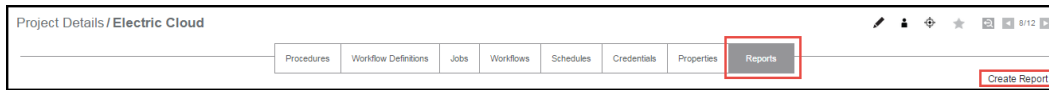
Note: For the Category, Count Over Time, and Multiple Series reports, a **View Search Results** link is available at the bottom of the report if a Saved Filter was used to generate the report.

Creating a Report Job

ElectricFlow runs reports as a job, so reports can be managed like any other job.

To create a report for your project:

1. Go to the Project Details page for your selected project.
2. Select the Reports subtab.
3. Click the **Create Report** link.



After clicking **Create Report**, the next screen is displayed with the default values for the selected tab (Multiple Series) in this example.

Multiple Series | Category | Count Over Time | Procedure Usage

Report Title: Multiple Series Report

Project: ==Select Project==

Saved Filter: Filter: [-None-]

Time Period: Yesterday

Create thumbnail? ☐

Object Type: Job

Chart Type	Function	Property Name	Display Name	Stacked
Line	Average	elapsedTime	Elapsed Time	<input type="checkbox"/>

Add Series

Chart Options

Table Options

Advanced

Run Report | Create Schedule | Cancel

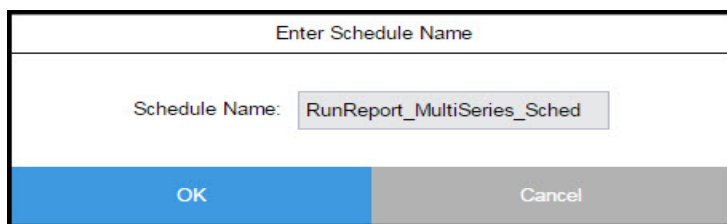
Enter information in the fields or select the appropriate values as follows:

- Report Title - Your report title. Type over the default report name, choosing any unique name for your report. We supplied Basic Build Report for our report name.
- Saved Filter:
 - Project - Click your mouse inside this field to see a list of projects from which to make a selection. We select our FirstNewProject project name.
 - Filter - Use the BasicBuildFilter we created earlier.
- Time Period - Use the drop-down menu to select the time period.
- Create thumbnail? - Check this box so you can view this report on your Home page.
(We will create a "thumbnail" report view on your Home page at the end of this scenario.)
- Object Type - Use the drop-down menu to select Job for this example.
- Table column choices:
 - Chart Type - Use the drop-down menu to choose the chart type.
 - Function - Use the drop-down menu to choose the function you need.
 - Property Name - Use the default property value or delete the text and click your mouse in the blank field to see a list of possible properties. We chose "outcome" for this example.
 - Display Name - You can choose a different unique Display Name if you prefer to do so.
 - Stacked - Select this check box to see your report results "stacked" versus overlaid.

- The "X" icon - Click this icon to delete any row you no longer need.
- Select the **Add Series** button if you would like to create additional table entries for additional report information.
- Chart Options - Use the down-arrow to adjust the Time Grouping and see the defaults for the X and Y axis.

Buttons at the bottom of the page:

- Run Report button - this button takes you to the Job Details page to run the report immediately—one time only.
- Click the **Create Schedule** button - this button displays a box with a default schedule name you can change.



- Enter a name for the schedule, **Basic Build Report Schedule**—again, we tied the schedule name to the procedure name for which we want the report.

Viewing the report

After the report job runs, view the report by selecting the Reports subtab within the project. **Note:** All reports are viewed from within a project context. If you have reports that span multiple projects, run them from the Electric Cloud project or create a project to store summary reports.

After selecting the Reports subtab, see the navigation tree on the left-side of the page. This tree allows you to select the report you want to view. The report tree contains three levels.

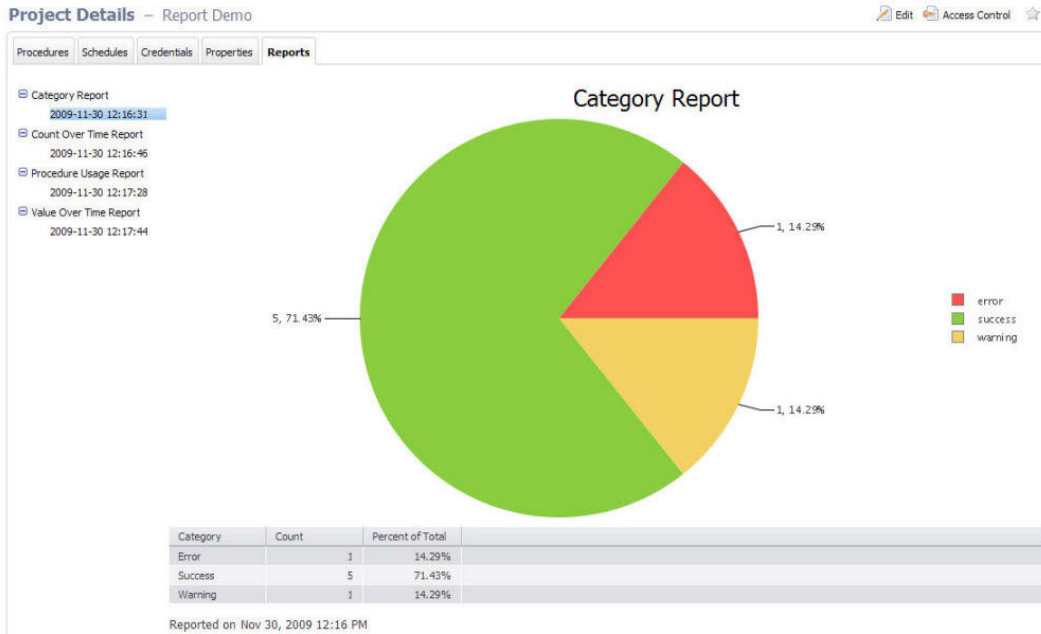
- Report Title - Click the "+" icon next to the report type to see available report versions.
- Report Date - Click this link to see report details.

Note: Click the report type to display the most recent report version.

Optional Batch report examples

Category sample report

This report shows the distribution of property values in a pie chart.



Creating this report on the Run Procedure page:

Run Procedure – RunReport_CategoryPie ☆

Parameters

Credential: Username: Password:

Locales:

Object Type:

Property: Default: "outcome"

Report Title: Default: "Category Report"

Saved Filter Project: ☒ Current ☐ Browse

Saved Filter Name: Browse

Time Period:

Advanced

Priority:

Impersonation: ☒ Use pre-defined credential ☐ Use specific credential ☐ Use a specific user

Field descriptions:

- **Credential** - these are credentials to be used when collecting data. If no credentials are specified (Username/Password remain blank), the report runs with the privileges of the project from where it is run. If credentials are supplied, the data is collected using the ElectricFlow user security access in the credential. Use credentials when you want to access data outside of your project.

- **Locales** -a comma separated list of locales where you would like to render the report. Locale is specified as one or more of the following: `en`, `zh`, `ja`, or `ko`, which translate respectively to English, Chinese, Japanese, or Korean. If no locale is specified, the default is to use the locale of the ElectricFlow server. For example, a server already set for Chinese will see reports in Chinese with no other changes required. Note: Currently, ElectricFlow supports four locales only.
- **Object Type** - the type of object to search (default job)
- **Property** - the property of interest (default outcome) and must be a valid property for the object type selected
- **Report Title** - your report title (default Category Report)
- **Saved Filter Project** - select "Current" or Browse to find the project where you want to save this filter.
- **Saved Filter Name** - the name of a saved filter to use when filtering collected data.

Use the Search tab to create the filter. After entering your search criteria, click **OK**.

If the preview displays the data you want, click **Save Filter** and provide a Project Name and Filter Name to specify where the filter is saved.

On the Run Report parameters page, select the filter. If you leave the Filter Name blank, no filtering is performed.

Note: "Saved filters" can filter on start and end times. Because start and end times are specified in the Time Period parameter, there is a possibility you could select a combination of filter date and time periods that exclude all data. For example, you could select a time period of "Yesterday" and a filter of Start=less than Jan 1, 1900. To avoid this problem, Electric Cloud recommends you do NOT include dates or times in your filter.

- **Time Period** - these are keywords to define the report start and end times. The range is used to filter collected data. All values are in the local time zone.
 - Today - midnight until now
 - Yesterday - previous full day
 - This week - from the start of the previous Sunday until now
 - Last week - from Sunday to Sunday of the last complete week
 - This month - from the start of the first of this month until now
 - Last month - from the last complete calendar month, which means on January 3rd the last full month is all of December
 - This year - from January 1st until now
 - Last year - all of last year (Jan 1 to December 31)
 - All - all dates until now

Procedure Usage Sample Report

For a specific procedure, this report shows procedures called by the procedure and which procedures it called.

Project Details – Report Demo

Edit Access Control Job Notes Access Control

Procedures Schedules Credentials Properties **Reports**

Category Report
2009-01-12 08:04:57.6
Count Over Time Report
2009-01-12 08:05:17.7
Procedure Usage Report
2009-01-12 08:17:05.0
Value Over Time Report
2009-01-12 08:07:14.9

Procedure Usage Report

Project	Procedure	Calls/Called By	Project	Procedure
My Library	LibProc-Bar	called by	My Library	LibProc-Foo
My Library	LibProc-Foo	calls	My Library	LibProc-Bar
My Library	LibProc-Foo	called by	Sample Project	Do Work

Reported on Jan 12, 2009 8:16 AM

Creating this report on the Run Procedure page:

Run Procedure – RunReport_ProcedureUsage

Parameters

Credential: Username: Password:

Locales:

Project: Browse

Project Procedure: Required; Default: "%"

Report Title: Default: "Procedure Usage Report"

Advanced

Priority:

Impersonation: ☒ Use pre-defined credential ☐ Use specific credential ☐ Use a specific user

Field descriptions:

- **Credential** - these are the credentials to use for collecting data. If no credentials are specified (Username/Password remain blank), the report runs with the privileges of the project from where it is run. If credentials are supplied, data is collected using the ElectricFlow user security access in the credential. Use credentials when you want to access data outside of your project.
- **Locales** - a comma separated list of locales where you would like to render the report. Locale is specified as one or more of the following: en , zh , ja , or ko, which translate respectively to English, Chinese, Japanese, or Korean. If no locale is specified, the default is to use the locale of the ElectricFlow server. For example, a server already set for Chinese will see reports in Chinese with no other changes required. **Note:** Currently, ElectricFlow supports four locales only.
- **Project** - this is the name of the project where you want to install the report.
- **Project Procedure** - enter a project procedure. "called" and "called by" data is shown for that project/procedure combination. You can use "wildcards." For example, to see data for ALL projects and procedures, enter the % character in each field.
- **Report Title** - this is the title of the report

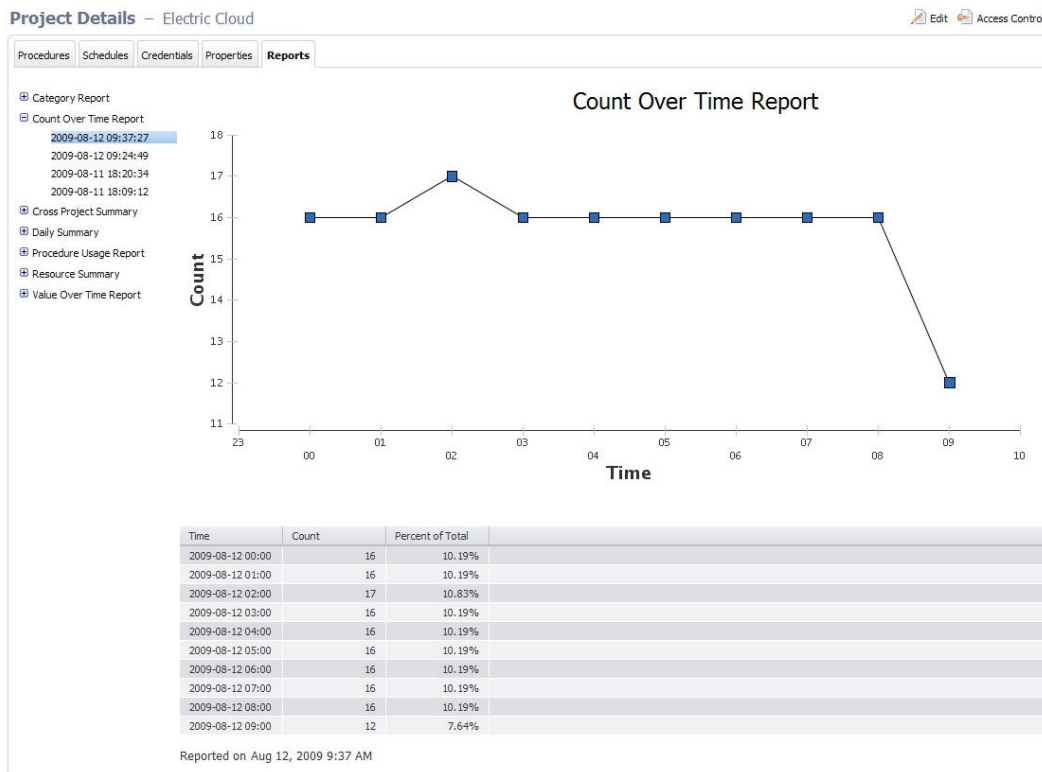
Count Over Time Sample Report

This report shows search results from counting multiple job, job step, or resource objects within the selected time range.

For example, this report counts the number of occurrences for the specified object-type over the selected time range--if a filter exists, it will be applied. This report does not aggregate the value (use ValueOverTime if you want aggregation), each occurrence counts as one.

Use this report when the property contains string values. The ValueOverTime report can be used only for numeric properties.

Note: Generating large numbers of datapoints can exceed BIRT's charting capability.



Creating this report on the Run Procedure page.

Run Procedure – RunReport_CountOverTime

Parameters

Chart Time Grouping: Default: "Time"

Chart Type: Default: "Count"

Chart X-Axis Label: Default: "Time"

Chart Y-Axis Label: Default: "Count"

Credential: Username: Password:

Locales:

Object Type: Default: "Count Over Time Report"

Report Title: Default: "Count Over Time Report"

Saved Filter Project: ☒ Current ☐ Browse

Saved Filter Name: Browse

Table Column Heading Property: Default: "Count"

Table Column Heading Time: Default: "Time"

Table Time Grouping:

Time Period:

Advanced

Priority:

Impersonation: ☒ Use pre-defined credential ☐ Use specific credential ☐ Use a specific user

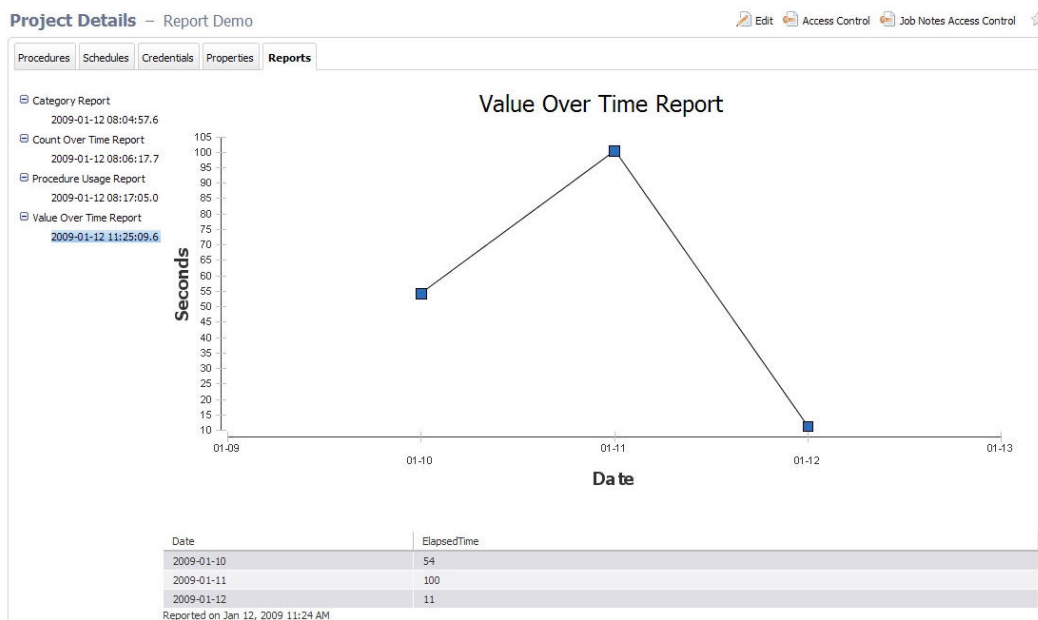
Field descriptions:

- Chart Time Grouping - these are: Auto (default), Minute, Hour, Day, Week, Month, or Year
- Chart Type - these are: Line (default), Bar, Cone, Tube, Triangle, or Area
- Chart X-Axis Label - the text to display on chart x-axis (default Date)
- Chart Y-Axis Label - the text to display on chart y-axis (default Count)
- Credential - credentials to be used when collecting data. If no credentials are specified (Username/Password remain blank), the report runs with the privileges of the project from where it is run. If credentials are supplied, the data is collected using the ElectricFlow user security access in the credential. Use credentials when you want to access data outside of your project.
- Locales - a comma separated list of locales where you would like to render the report. Locale is specified as one or more of the following: en, zh, ja, or ko, which translate respectively to English, Chinese, Japanese, or Korean. If no locale is specified, the default is to use the locale of the ElectricFlow server. For example, a server already set for Chinese will see reports in Chinese with no other changes required. Currently, ElectricFlow supports four locales only.
- Object Type - the type of object to search (default job)
- Report Title - your report title (default Count Over Time Report)
- Saved Filter Project - select "Current" or Browse to find the project where you want to save this filter.
- Saved Filter Name - the name of a saved filter to use when filtering collected data.//Use the Search tab to create the filter. After entering your search criteria, click **OK**.//If the preview displays the data you want, click **Save Filter** and provide a Project Name and Filter Name to specify where the filter is saved.// On the Run Report parameters page, select the filter. If you leave the Filter Name blank, no filtering is performed.//**Note:** "Saved filters" can filter on start and end times. Because start and end times are specified in the Time Period parameter, there is a possibility you could select a combination of filter date and time periods that exclude all data. For example, you could select Time Period of "Yesterday" and a filter of Start=less than Jan 1, 1900. To avoid this problem, Electric Cloud recommends you do NOT include dates or times in your filter.

- Table Column Heading Property - the table heading for the right column (default Count)
- Table Column Heading Time - the table heading for the left column (default Date)
- Table Time Grouping - these are: Auto (default), Minute, Hour, Day, Week, Month, Year
- Time Period - these are keywords to define the report start and end times. The range is used to filter collected data. All values are in the local time zone.
 - Today - midnight until now
 - Yesterday - previous full day
 - This week - from the start of the previous Sunday until now
 - Last week - from Sunday to Sunday of the last complete week
 - This month - from the start of the first of this month until now
 - Last month - from the last complete calendar month, which means on January 3rd the last full month is all of December
 - This year - from January 1st until now
 - Last year - all of last year (Jan 1 to December 31)
 - All - all dates until now

Multiple Series Report

This reports shows the cumulative value of a property over time. This report should be run against properties that contain numbers. (Use the Count Over Time report if your property contains string values.) You can specify the function to use for aggregating values (sum, count, average). For example: Assume a property that contains the number of failed unit tests is stored on each job. Using the Multiple Series report, you can see the failure trend over time.



Creating this report on the Run Procedure page:

Run Procedure – RunReport_ValueOverTime

Parameters

Chart Time Grouping: Default: "Auto"

Chart Type: Default: "Line"

Chart X-Axis Label: Default: "Date"

Chart Y-Axis Label: Default: "Seconds"

Credential: Username: Password:

Locales:

Object Type: Default: "Job"

Property: Default: "elapsedTime"

Property Expression:

Property Function: Default: "Average"

Report Title: Default: "Value Over Time Report"

Saved Filter Project: ☒ Current ☐ Browse

Saved Filter Name: Browse

Table Column Heading Property:

Table Column Heading Time: Default: "Date"

Table Time Grouping:

Time Period:

Advanced

Priority:

Impersonation: ☒ Use pre-defined credential ☐ Use specific credential ☐ Use a specific user

Field descriptions:

- Chart Time Grouping - these are: Auto (default), Minute, Hour, Day, Week, Month, or Year
- Chart Type - these are: Line (default), Bar, Cone, Tube, Triangle, or Area
- Chart X-Axis Label - the text to display on chart x-axis (default Date)
- Chart Y-Axis Label - the text to display on chart y-axis (default Count)
- Credential - these are credentials to be used when collecting data. If no credentials are specified (Username/Password remain blank), the report runs with the privileges of the project from where it is run. If credentials are supplied, the data is collected using the ElectricFlow user security access in the credential. Use credentials when you want to access data outside of your project.
- Locales - a comma separated list of locales where you would like to render the report. Locale is specified as one or more of the following: en, zh, ja, or ko, which translate respectively to English, Chinese, Japanese, or Korean. If no locale is specified, the default is to use the locale of the ElectricFlow server. For example, a server already set for Chinese will see reports in Chinese with no other changes required. **Note:** Currently, ElectricFlow supports four locales only.
- Object Type - the type of object to search (default job)
- Property - the property of interest (default outcome) and must be a valid property for the object type selected
- Property Expression - an optional mathematical function to apply to your data. For example, elapsed time is recorded in milliseconds. To make a useful report with elapsed times, you may want to show seconds, so this field would be entered as "/1000".

- Property Function - the following list contains the aggregation functions to use for data in the chart and table.

Count	Median
Count Distinct	Mode
First	Std Deviation
Last	Sum
Min	

- Report Title - your report title (default Value Over Time Report)
- Saved Filter Project - select "Current" or Browse to find the project where you want to save this filter.
- Saved Filter Name - the name of a saved filter to use when filtering collected data.

Use the Search tab to create the filter. After entering your search criteria, click **OK**.

If the preview displays the data you want, click **Save Filter** and provide a Project Name and Filter Name to specify where the filter is saved.

On the Run Report parameters page, select the filter. If you leave the Filter Name no filtering is performed.

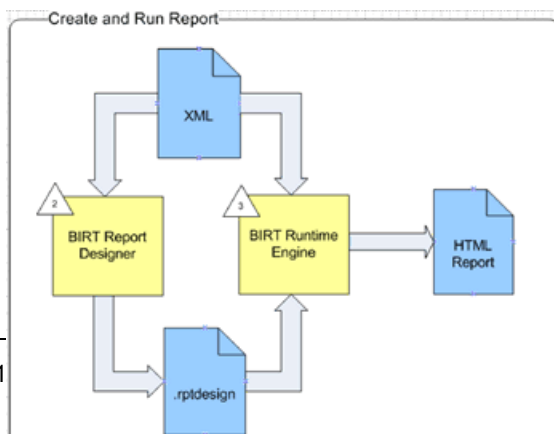
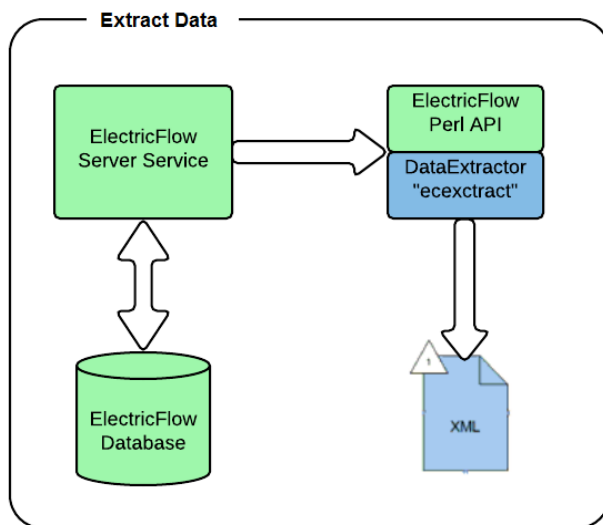
Note: "Saved filters" can filter on start and end times. Because start and end times are specified in the Time Period parameter, there is a possibility you could select a combination of filter date and time periods that exclude all data. For example, you could select Time Period of "Yesterday" and a filter of Start=less than Jan 1, 1900. To avoid this problem, Electric Cloud recommends you do NOT include dates or times in your filter.

- Table Column Heading Property - the table heading for the right column (default Count)
- Table Column Heading Time - the table heading for the left column (default Date)
- Table Time Grouping - these are: Auto (default), Minute, Hour, Day, Week, Month, Year

- Time Period - these are keywords to define the report start and end times. The range is used to filter collected data. All values are in the local time zone.
 - Today - midnight until now
 - Yesterday - previous full day
 - This week - from the start of the previous Sunday until now
 - Last week - from Sunday to Sunday of the last complete week
 - This month - from the start of the first of this month until now
 - Last month - from the last complete calendar month, which means on January 3rd the last full month is all of December
 - This year - from January 1st until now
 - Last year - all of last year (Jan 1 to December 31)
 - All - all dates until now

Creating Custom reports

The ElectricFlow batch mode reports which, by default, run once each night and are accessed through the Reports tab [in the Electric Cloud project] in the web UI. These reports are built using a combination of the ElectricFlow API and the BIRT Report Designer tool. BIRT is an open source Eclipse-based reporting system you can use to create your own custom reports, or modify existing ElectricFlow reports. This section describes how to write your own custom report and integrate it into ElectricFlow.



There are several steps to using BIRT to create reports:

- Create an XML file with data using the ElectricFlow Perl API
- Design a report using BIRT RCP Report Designer
- Run the report with the BIRT Report Engine
- View the report from within ElectricFlow

Data extraction

To begin creating a custom report, you must first extract report data from ElectricFlow and move it into a standalone XML file. The recommended way to create the XML file is to use the ElectricFlow `ecextract` utility program, which is derived from the Perl API. The benefits of using `ecextract` to create XML files are:

- **Security** - `ecextract` enforces all ElectricFlow security and ACLs. (Direct DB access does not honor ACLs.)
- **Maintainability** - `ecextract` is supported by Electric Cloud, including backward compatibility and insulates you from database schema changes.
- **Simplicity** - `ecextract` provides easy-to-use search and filtering options. Because of the complex nature of our schema, particularly for properties, it may be difficult to create custom SQL statements.
- **Re-use** - You may find that XML files created by built-in reports already contain all the data you need. If you simply want to present data differently or "slice" the data in a different way, you can re-use the XML from the existing built-in reports.

Note: If you already use `ectool` or Perl for data extraction, and have already created the scripts you need for custom reports, you may continue to use those scripts. However, Electric Cloud recommends using `ecextract` if you are new to creating custom reports.

Using `ecextract.pl` for data extraction

A Perl program, `ecextract.pl`, is provided so you do not have to write Perl code—or you can use it as a starter for your own Perl API calls. This script allows you to specify the most common query types on a command line and generates an XML file. The following table lists currently available `ecextract` arguments:

Usage: `ecextract [options]`

Data Extract Options:	
<code>--type job jobStep resource</code>	The object type to query.
<code>--property [label=]property</code>	Add a property to the output. If label is specified, it is used as the XML tag name instead of the property name.
<code>--filter property, operator, operand1 [,operand2]</code>	Add a property filter to the query.

<code>--savedfilter filterName</code>	Use a saved filter (ignores <code>--filter</code>)
<code>--period period</code>	<p>The time period. One of: today, yesterday, thisweek, lastweek, thismonth, lastmonth, thisyear, lastyear, all, window, <number of days>, or range,<start-time>,<end-time></p> <p>Range start, end time is date in ISO 8601 format: YYYY-MM-DDTHH:MM:SS[Z]If the date ends with 'Z', it is considered to be GMT time, otherwise the local timezone is used.</p>
<code>--sort property, order</code>	Add a sort to the query. Order is either 'ascending' or 'descending'.
<code>--outputPath</code>	Path and name of output file. Default is <code>out.xml</code> .
<code>--max</code>	Maximum number of objects to return.
Server Communication Options:	
<code>--server server</code>	Address for the ElectricFlow server. Defaults to the value of the <code>COMMANDER_SERVER</code> environment variable. If it does not exist, defaults to localhost.
<code>--port port</code>	HTTP listener port on the server. Defaults to 8000.
<code>--securePort port</code>	HTTPS listener port on the server. Defaults to 8443.
<code>--secure 0 1</code>	If set to true, HTTPS is used to communicate with the server. Defaults to true.

<code>--timeout seconds</code>	Set the number of seconds for the timeout. Defaults to 280.
Login Options:	
<code>--user user</code>	ElectricFlow login user.
<code>--pass password</code>	Password for login user.
<code>--credentialName</code>	Credential name (valid only when running inside a job step).
Global Options:	
<code>--help</code>	Print this message and exit without doing anything.
<code>--version</code>	Debug level. Higher values... more details 0 - errors only 1 - 0 + System progress 2 - 1 + Property Warnings 3 and higher give detailed diagnostics
Plugin File Options:	
<code>--load loadFile</code>	Plugin perl program to execute
<code>--</code>	Stop processing ecextract options. Allows plugins to process their own options following this option.

`ecextract.pl` is part of the EC-Reports plugin—this script is located in the "`agent/bin`" directory of the EC-Reports plugin.

Notes:

`--property`

Properties can be specified as high-level values (`--property propertyName`), properties in a sheet (`--property sheet1/sheet2/propertyName`), or as relative paths (`--property /myProject/propertyName`).

`--filter`

Each entry consists of a property name to test and an operator to use for comparison. The property can be an intrinsic property defined by ElectricFlow or a custom property added by the user. Each operator takes zero or one operand to compare against the desired property.

The operators are:

contains, equals, greaterOrEqual, greaterThan, in, lessOrEqual, lessThan, like, notEqual, notLike, isNotNull, isNull

Example: `--filter jobName,equals,Hello`

Examples

The following 3 examples contain the script input and the output.

Example 1: Find properties on job steps for one day in April

```
ec-perl ecextract.pl ^
--user report ^
--pass report ^
--server localhost ^
--outputPath jobStepOut.xml ^
--type jobStep ^
--max 1000 ^
--property jobId ^
--property finish ^
--property jobStepId ^
--property stepName ^
--property status ^
--property outcome ^
--property runTime ^
--property waitTime ^
--filter jobName,like,commander-2.2%% ^
--filter createTime,greaterOrEqual,2008-04-01T00:00:00.000Z ^
--filter createTime,lessOrEqual,2008-04-02T00:00:00.000Z ^
--sort jobId,descending
```

Output from example 1

```
<?xml version="1.0" encoding="utf-8"?>
<response>
  <object objectType="jobStep" objectId="jobStep-3511785">
    <jobId name="jobId">4fa765dd-73f1-11e3-b67e-b0a420524153</jobId>
    <finish name="finish">2009-04-01T23:54:45.424Z</finish>
    <jobStepId name="jobStepId">4fa765dd-73f1-11e3-b67e-b0a420524153</jobStepId>
    <stepName name="stepName">Setup</stepName>
    <status name="status">completed</status>
    <outcome name="outcome">success</outcome>
```

```

    <runTime name="runTime">0</runTime>
    <waitTime name="waitTime">922</waitTime>
  </object>
  <object objectType="jobStep" objectId="jobStep-3511915">
    <jobId name="jobId">4fa765dd-73f1-11e3-b67e-b0a420524153</jobId>
    <finish name="finish">2009-04-01T23:59:26.844Z</finish>
    <jobStepId name="jobStepId">5da765dd-73f1-11e3-b67e-b0a420524153</jobStepId>
    <stepName name="stepName">Drop</stepName>
    <status name="status">completed</status>
    <outcome name="outcome">success</outcome>
    <runTime name="runTime">0</runTime>
    <waitTime name="waitTime">266</waitTime>
  </object>
  .....
</response>

```

Example 2: Find all jobs that were created in April 2008

```

ec-perl ecextract.pl ^
--user report ^
--pass report ^
--server localhost ^
--outputPath jobOut.xml ^
--type job ^
--max 100 ^
--property jobName ^
--property jobId ^
--property projectName ^
--property procedureName ^
--property branch ^
--filter createTime,greaterOrEqual,2009-04-01T00:00:00.000Z ^
--filter createTime,lessOrEqual,2009-04-30T00:00:00.000Z ^
--sort projectName,ascending ^
--sort procedureName,ascending ^
--sort createTime,descending

```

Output from example 2

```
<?xml version="1.0" encoding="utf-8"?>
<response>
  <object objectType="job" objectId="job-158461">
    <jobName name="jobName">commander-3.2.23916-200904011650</jobName>
    <jobId name="jobId">4fa765dd-73f1-11e3-b67e-b0a420524153</jobId>
    <projectName name="projectName">Commander</projectName>
    <procedureName name="procedureName">Master</procedureName>
    <branch name="branch">3.2</branch>
  </object>
  <object objectType="job" objectId="job-158455">
    <jobName name="jobName">commander-anders-main-ad.23915-200904011627</jobName>
    <jobId name="jobId">4fa765dd-73f1-11e3-b67e-b0a420524153</jobId>
    <projectName name="projectName">Commander</projectName>
    <procedureName name="procedureName">Master</procedureName>
    <branch name="branch">main</branch>
  </object>
  .....
</response>
```

Example 3: Find all resources

```
ec-perl ecextract.pl ^
--user report ^
--pass report ^
--server localhost^
--outputPath resourceOut.xml ^
--type resource ^
--max 100 ^
--property resourceName ^
--property stepCount ^
--property stepLimit ^
--property hostName ^
--filter stepCount,greaterThan,0 ^
--sort resourceName,ascending
```

Output from example 3

```
<?xml version="1.0" encoding="utf-8"?>
<response>
```

```

<object objectType="resource" objectId="resource-106">
  <resourceName name="resourceName">ecbuild-lin1</resourceName>
  <stepCount name="stepCount">2</stepCount>
  <stepLimit name="stepLimit">0</stepLimit>
  <hostName name="hostName">ecbuild-lin1</hostName>
</object>
<object objectType="resource" objectId="resource-108">
  <resourceName name="resourceName">ecbuild-win1</resourceName>
  <stepCount name="stepCount">1</stepCount>
  <stepLimit name="stepLimit">0</stepLimit>
  <hostName name="hostName">ecbuild-win1</hostName>
</object>
.....
</response>

```

BIRT Report Designer

Now that you have a dataset, you can create a BIRT report. You need an XML file to use while designing the report. If you did not create an XML data file yet, now is the time to do so. After you have sample data, you are ready to start designing a report.

Report definitions are created with the BIRT Report Designer, currently version 2.5.2. These definitions are then stored in `.rptdesign` files. You have several options for installing the designer tool:

- Use a BIRT Report Designer bundled and already installed with Eclipse, or
- download BIRT Report Designer to use with your existing Eclipse installation, or
- because Eclipse is not required, you can download only the BIRT Report Designer (this is the easiest and preferred method).

Electric Cloud recommends downloading the BIRT RCP Report Designer only, which is the standalone design tool. The url for downloading the Report Designer is:

http://www.eclipse.org/downloads/download.php?file=/birt/downloads/drops/R-R1-2_5_2-201002221500/birt-rcp-report-designer-2_5_2.zip

The general download page for BIRT 2.5.2 with the other options discussed is
http://download.eclipse.org/birt/downloads/build.php?build=R-R1-2_5_2-201002221500

Using the BIRT Report Designer is beyond the scope of this Help topic. Documentation and tutorials can be found on the BIRT website: <http://www.eclipse.org/birt>. You may find it helpful to view the report designs for built-in ElectricFlow. These designs can be found in the `agent/reports/<report name>` directory in the EC-Reports plugin.

Understanding additional report components

In addition to the XML dataset and report design, there are optional report components you may wish to include in your report.

- **External style sheet**
BIRT supports external style sheets for reports. If you use these style sheets, they need to be located with your report design. For example, built-in ElectricFlow reports use an external style sheet file "styles_birt.css".
- **Externalized string properties**
String literals (used in your report design) can be externalized into property files to support localization. These property files need to be located with your report design also. The properties file for the default locale will have the same name as your report and a ".properties" file extension, for example, `CategoryPie.properties`. Text for additional locales are stored in files named `<report name>_<locale>.properties`, for example, `CategoryPie_ja.properties` contains text localized for the Japanese language.

You may review examples of these additional components for built-in ElectricFlow reports. These files are located in the EC-Reports plugin, in the `agent/reports/<report name>` directory.

Packaging reports for ElectricFlow

After you have an XML dataset (using `ecextract.pl`), a BIRT report design (using BIRT Report Designer), and any additional optional report components completed, you are ready to integrate your report into ElectricFlow. To do this:

- Store all the report components in a location accessible to your ElectricFlow agent—you can use the ElectricFlow plugins directory. For example, for your report files, you could create a directory called "MyReport" in the ElectricFlow plugins directory.
- Create an ElectricFlow procedure to perform the necessary steps to generate a report, including: extracting data, running the report design, and registering report output to appear on the Report tab on the Project Details page. You may want to copy one of the procedures from the built-in ElectricFlow reports to use as a template. The following command creates a copy of the `RunReport_CategoryPie` procedure called `RunReport_MyReport` in the project of your choice. Substitute your project name for `<my-project-name>` in the following command.

```
ectool clone --cloneName/projects/<my-project-name>/procedures/RunReport_
MyReport
--projectName/plugins/EC-Reports/project --procedureName RunReport_
CategoryPie
```

Example: customizing the command body

The following is an example of customizing the procedure contents for the `RunReport_MyReport` procedure, cloned from the `RunReport_CategoryPie` procedure above.

One customization is required and two are optional. The remainder of the command body does not need to be changed.

- **Customizing the location of the BIRT report design file**
Required: update the location of the BIRT report design file.

- **Original value:** our \$reportDesignFile =
"\$pluginDir/agent/reports/CategoryPie/CategoryPie.rptdesign";
- **New value:** our \$reportDesignFile = "\$ENV{'COMMANDER_PLUGINS'}
/MyReport/MyReport.rptdesign";
- **Customizing data extraction**
To customize data extraction using `ecextract.pl`, modify the following section of the command body:

```
#-----
# extractReportData
#
#      Extract report data from Commander using ecextract utility
#-----

sub extractReportData()
{
    # Time property varies based on object type
    my $timeprop = "modifyTime";
    if (
        $timeprop = "finish";
    )
    my $command = "ec-perl \"$pluginDir/agent/bin/ecextract.pl\" \"
        . \" --debug 3\"
        . \" --outputPath \"$ ecextractXmlFile\"\"
        . \" --credential \"${Credential}\"\"
        . \" --type \"${Object Type}\"\"
        . \" --property time=$timeprop\"
        . \" --property series1=\"${Property}\"\";
    if ("${Saved Filter}" ne "") {
        command = $command . \" --savedFilter \"${Saved Filter}
\"\"

    }
    $command = $command . \" --period \"${Time Period}\"\";
    print "data extract [$command]\n";
    my $result = ` $command `;
    print "extract result:\n$result";
    if ($?) {
        exit (1);
    }
}
```

- Customizing report parameters

To customize the parameters passed to your report design, modify the following section of the command body:

```
#-----  
# getReportArgs  
#  
#      Get report arguments  
#-----  
  
sub getReportArgs() {  
    # args for creating report  
    my $categoryHeading = ucfirst($[Property]);  
    my @args = (  
        "-f", "$reportFormat",  
        "-o", "$reportName",  
        "-i", "images/",  
        "-p", " xmlfile=../$ecextractXmlFile",  
        "-p", "ReportTitle=$reportTitle",  
        "-p", "TableColHeadingTime=$categoryHeading",  
        "-p", "TableColHeadingSeries1=Count" ,  
        "-p", " SearchURL=" . getSearchUrl(),  
        "$reportDesignFile",  
    );  
  
    # add locale args  
    foreach my $locale (@locales) {  
        if ($locale ne "") {  
            unshift @args, "-l $locale";  
        }  
    }  
    return @args;  
}
```

Helper functions provided in ElectricCommander::ReportUtils.pm

ecgenReport - runs the BIRT report, using the BIRT Report Designer

- **installationDirectory** - this is the directory where ElectricFlow is installed and where you will find JAVA and BIRT
- **artifactDirectory** - this is the directory where reports, images, and other artifacts should be stored. To display using the Reports subtab, these items must be in a directory named "artifacts" at the top of the job's workspace
- **args** - an array of arguments to the BIRT runtime engine, which specifies parameters, design templates, and so on...

registerReport - marks the report so it will be displayed when you select the Reports subtab

- **commander** - an `ElectricCommander()` object used to make ElectricFlow calls. If you need to use a specific user context, call the `login` method before calling `registerReport`
- **jobId** - this is the report job ID, which is used to set properties on the job
- **url** - this is the URL to register and it takes the form workspace name/html file name
- **title** - this is the text string for the report navigation tree
- **treeGroupings** - a hash grouped `name/values` set as properties to define the report tree groupings—report tree groupings are defined on a propertySheet named `ec_reportConfiguration` on the project or on the server
 - **hash key** - this is a property name that will be created on job's `ec_reportData/$title/` property sheet and used for report tree grouping
 - **hash value** - property value

registerArtifactsDirectory - creates the property required to set the artifacts directory

- **commander** - an `ElectricCommander()` object used to make ElectricFlow calls
- **jobId** - this is the report job ID, which is used to set properties on the job
- **artifactsDir** - this is a directory path relative to the workspace where job artifacts will be stored

extractFile - this function pulls content from the report definition (stored in properties) and creates files from them

- **commander** - an `ElectricCommander()` object used to make ElectricFlow calls. If you need to use a specific user context, make sure to call the `login` method before calling `registerReport`
- **jobId** - this is the report job ID, which is used to set properties on the job
- **reportType** - this is used to lookup report attributes
- **files** - an array of hashes that specify which properties should be loaded into which files
 - **prop** - the property name from the installed report type
 - **dest** - the destination file name (relative to the current working directory)
 - **expand** - this is "1" if you want ElectricFlow to expand property references in the property, "0" if not

getReportTimestamp - return hires timestamp formatted in a specified time zone

- **TimeZone** - (optional) a time zone specifier, default is "local"
- **FormatString** - (optional) a format specifier for the timestamp string

Custom Report Examples

This Help topic contains two examples illustrating how you might use the BIRT Report Designer to modify a built-in ElectricFlow report to create a custom report more meaningful for your purposes.

Example 1: modifying an existing report - adding a "banner" heading

The easiest way to customize a report is to take an existing report and modify it. ElectricFlow provides a set of built-in reports you can modify.

Process overview for modifying an ElectricFlow built-in report

- Make a copy of an existing report
- Make changes to the report
- Test your new custom report

To copy an existing ElectricFlow report

For this example, we will make a copy of the built-in Value Over Time report. The process to copy a report is:

- Make a copy of the file system report components of the Value Over Time report located in the ElectricFlow plugins directory.

Rename the copied directory to MyReport.

The default location of the ElectricFlow plugins directory is:

On Linux: `/opt/electriccloud/electriccommander/plugins`

On Windows: `C:\Documents and Settings\All Users\Application Data\Electric Cloud\ElectricCommander\plugins`

Linux example - The following commands will make a copy of the Value Over Time report file components and then rename the files:

```
$ cd /opt/electriccloud/electriccommander/plugins
$ cp -r EC-Reports-1.0.0.*/agent/reports/ValueOverTime/ MyReport
$ cd MyReport
$ rename ValueOverTime MyReport ValueOverTime*
```

The contents of the MyReport directory should be:

```
$ ls -l
MyReport_en.properties
MyReport_ja.properties
MyReport_ko.properties
MyReport.properties
MyReport.rptdesign
MyReport_zh.properties
```

```
styles_birt.css
```

- Make a copy of the RunReport_ValueOverTime procedure from the EC-Reports plugin that performs the steps to generate a report, including extracting data, running the report design, and registering the report output to appear on the Report tab of the Project Details page. In this example, we will create a Project called “MyReportProject” and the procedure we copy will be called “RunReport_MyReport”.

```
$ ectool createProject MyReportProject

$ ectool clone --cloneName /projects/MyReportProject /procedures/RunReport_
MyReport

--projectName /plugins/EC-Reports/project --procedureName RunReport_
ValueOverTime

response requestId="1"

<cloneName>RunReport_MyReport</cloneName>

</response>
```

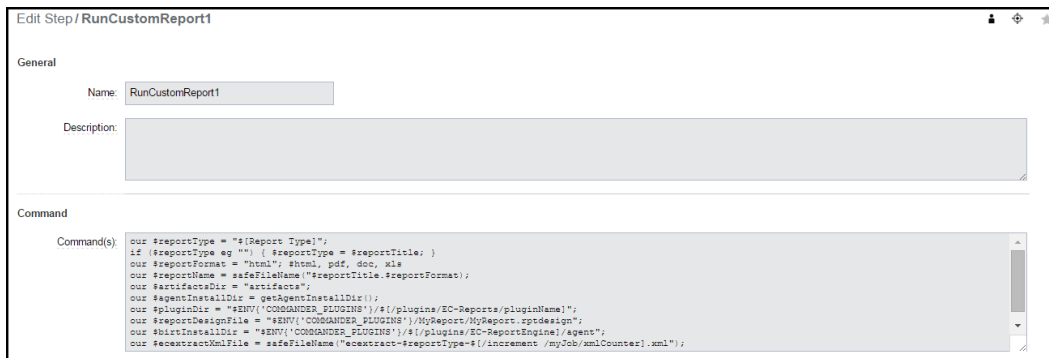
- Edit the “RunReport_MyReport” procedure in “MyReportProject” to reference the report design file copied earlier.
- Navigate to project MyReportProject, procedure RunReport_MyReport and select the RunCustomReport step to edit the Command block:

Find the following line:

```
our $reportDesignFile =
"$pluginDir/agent/reports/ValueOverTime/ValueOverTime.rptdesign";
```

and change this line to:

```
our $reportDesignFile = "$ENV{'COMMANDER_PLUGINS'}/ MyReport/MyReport.rptdesign";
```



Click **OK** to save the change.

To modify the copied report design

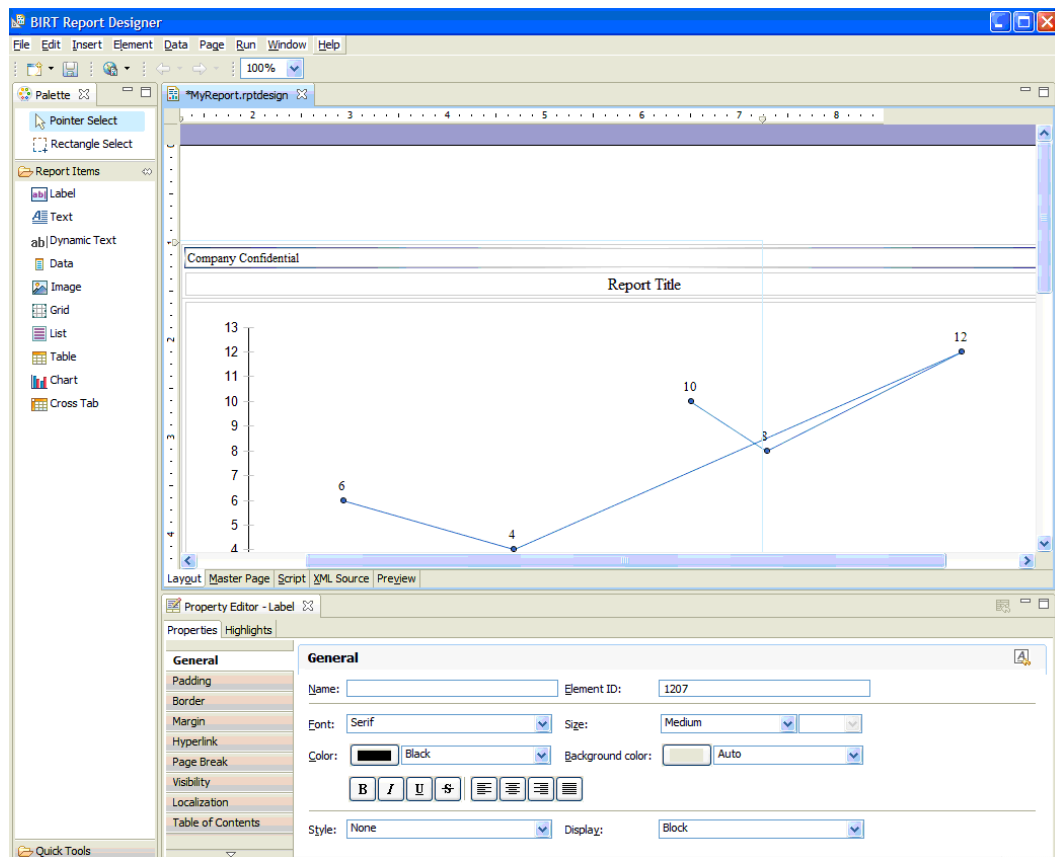
Start the BIRT Report Designer and open the report design file (MyReport.rptdesign) located in the MyReport directory previously created in the ElectricFlow plugins directory.

At this point... You are now ready to begin editing an existing report to create a custom report.

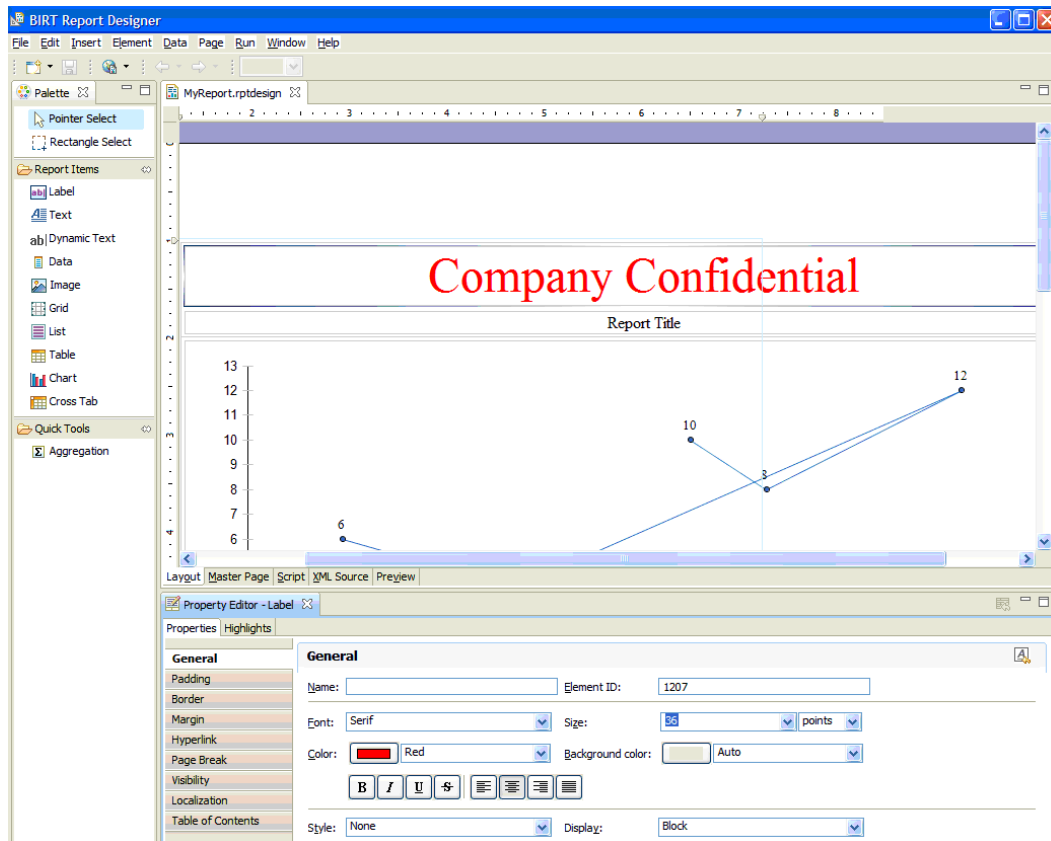
For our example, we are adding a new heading, "Company Confidential," (in large red letters) to the top of the existing report.

1. On the BIRT screen, select the Layout tab.
2. Drag a "Label" from the Palette window into the top of the report and type "Company Confidential" (or whatever text you would like to use for your test report) into the Label box.

See the next BIRT screen example.



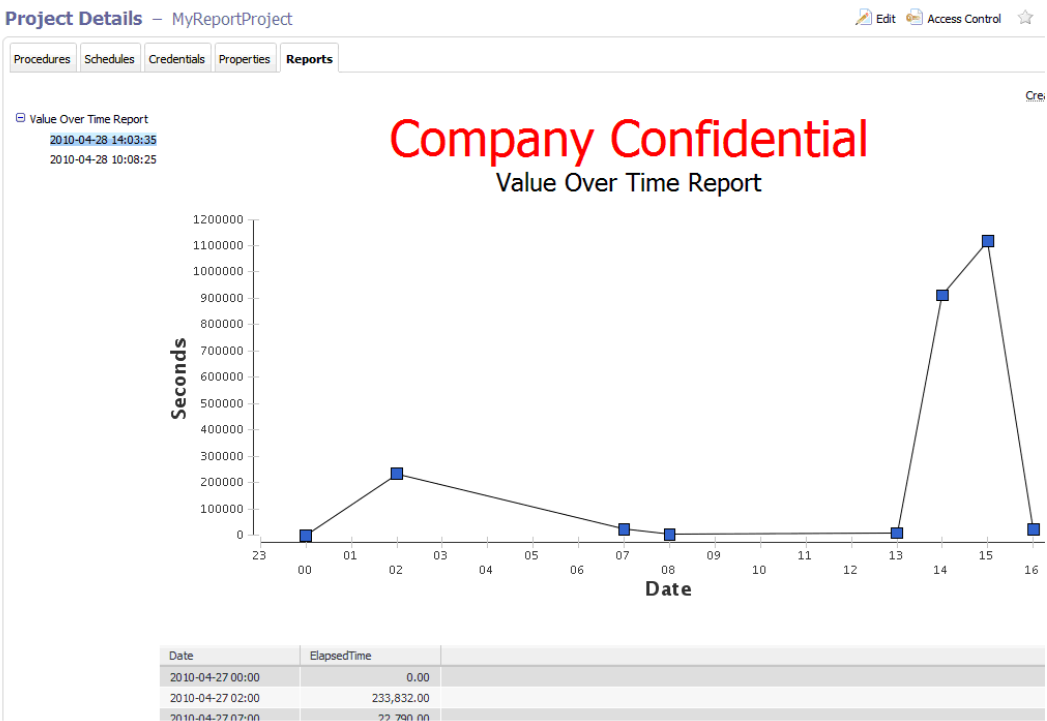
3. Now you can change the font size and color to create a banner appearance. Select the Property Editor window to customize your text - see the next screen example.
4. After customizing your text, Save your report.



Congratulations! You have just created a custom report.

Test your new report

Run the report by running the RunReport_MyReport procedure in the MyReportProject project. When the job completes, your new report will be available in the Links section on the Job Details page and from the Reports tab if the MyReportProject project. You will see the new header in the report output.



Example 2: complete end-to-end example

Using BIRT and some scripting expertise, you can create almost any kind of report you can imagine. This example creates a new report that shows the runtime trend of all job steps. Tasks for creating this report can be grouped into the following steps:

- Plan your report
- Create a new BIRT rptdesign file
- Deploy your custom report
- Test your new report

Planning this report example

We decided our report will be a single-series type report that will show job steps run time. The report will include a line chart at the top of the page whose X and Y axis represent the number of seconds and the job step ID respectively. We also want a table with two columns—column names will be "Job Step Name" and "Run Time".

Next, we need to gather all materials:

Generate sample data to test in the report.

You can use data from a previous run of a Value Over Time report—this file is located in the workspace of a previous reporting job and has the name eextract.xml.

Also from the copy of an ElectricFlow Value Over Time report, we will "copy and paste" a few values from it to save some typing.

Note: This example briefly describes items we are working with, but for more information about BIRT, you may need to consult BIRT documentation.

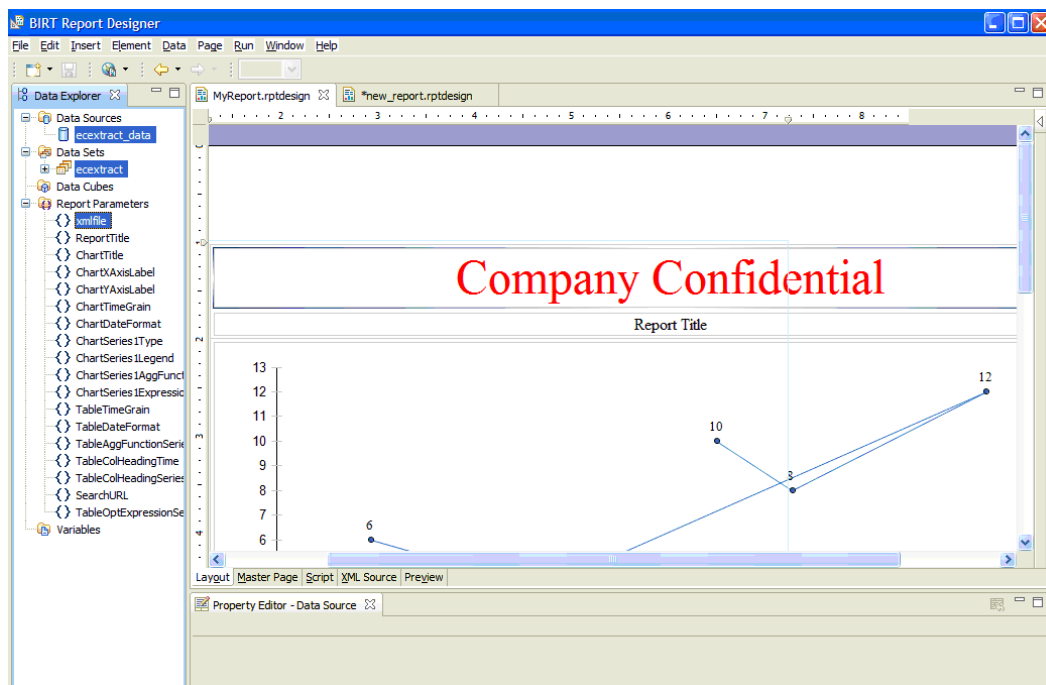
Creating a new BIRT rptdesign file

In BIRT Report Designer, create a new rptdesign file. Open the report file you created in "Example 1: modifying an existing report - adding a banner heading" so you can use the previous example report to copy some items into your new report.

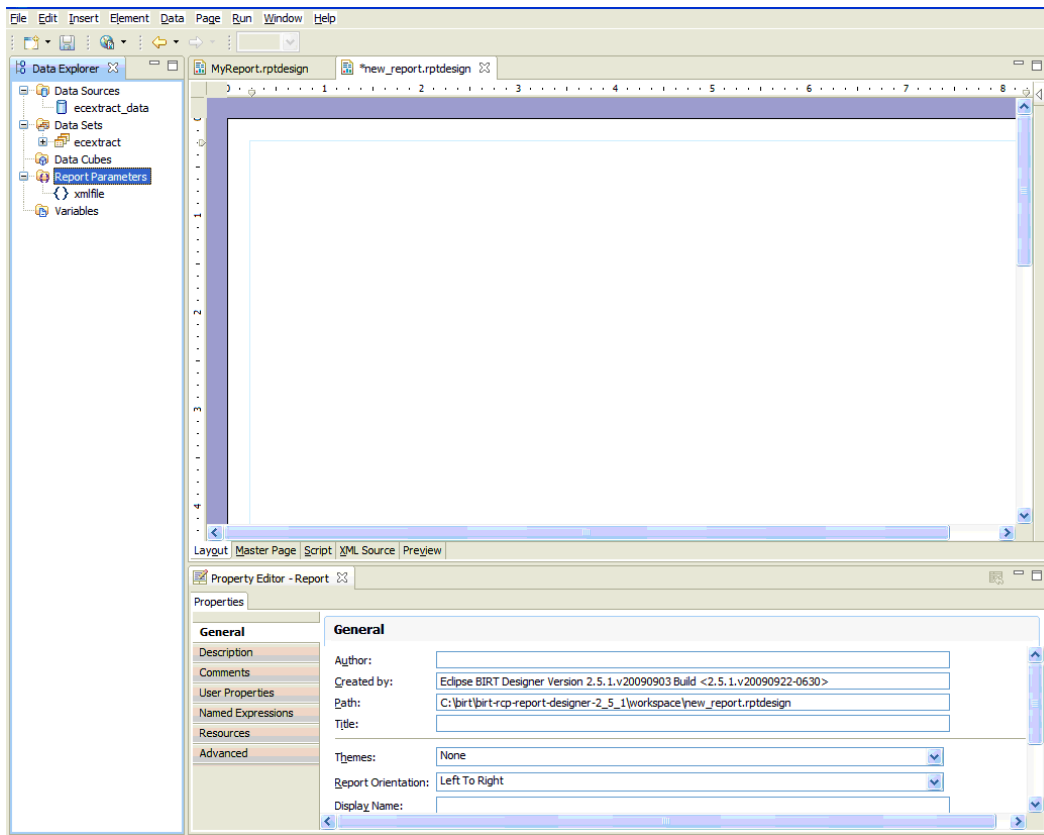
Items to copy into the new report:

- From Data Sources, copy ecextract_data
- From Data Sets, copy ecextract
- From Report Parameters, copy xmlfile

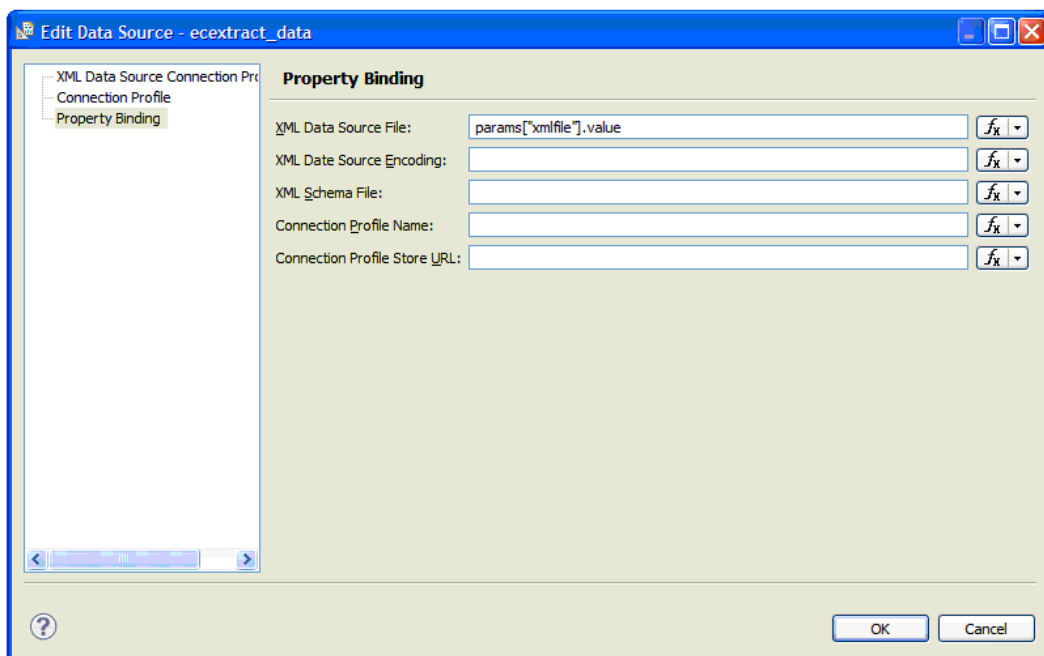
See the next screen example to help find these items.



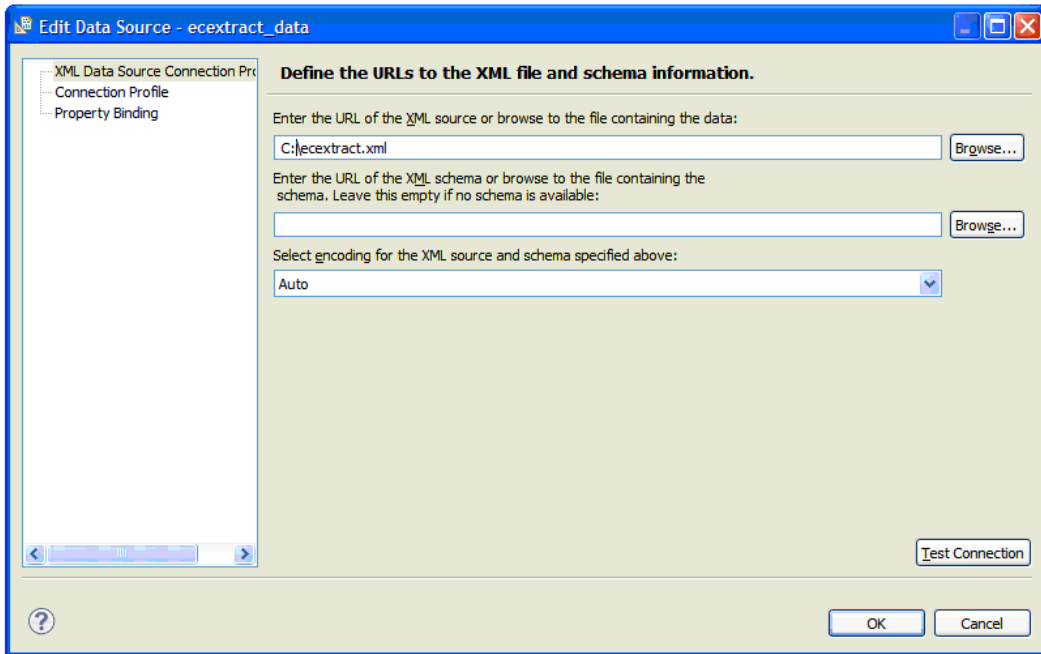
After copying these items, your new report design will look like the following example:



Double-click the Data Sources > ecextract_data file to ensure the XML file is referenced as a parameter. If it is not, type in the value as shown below.

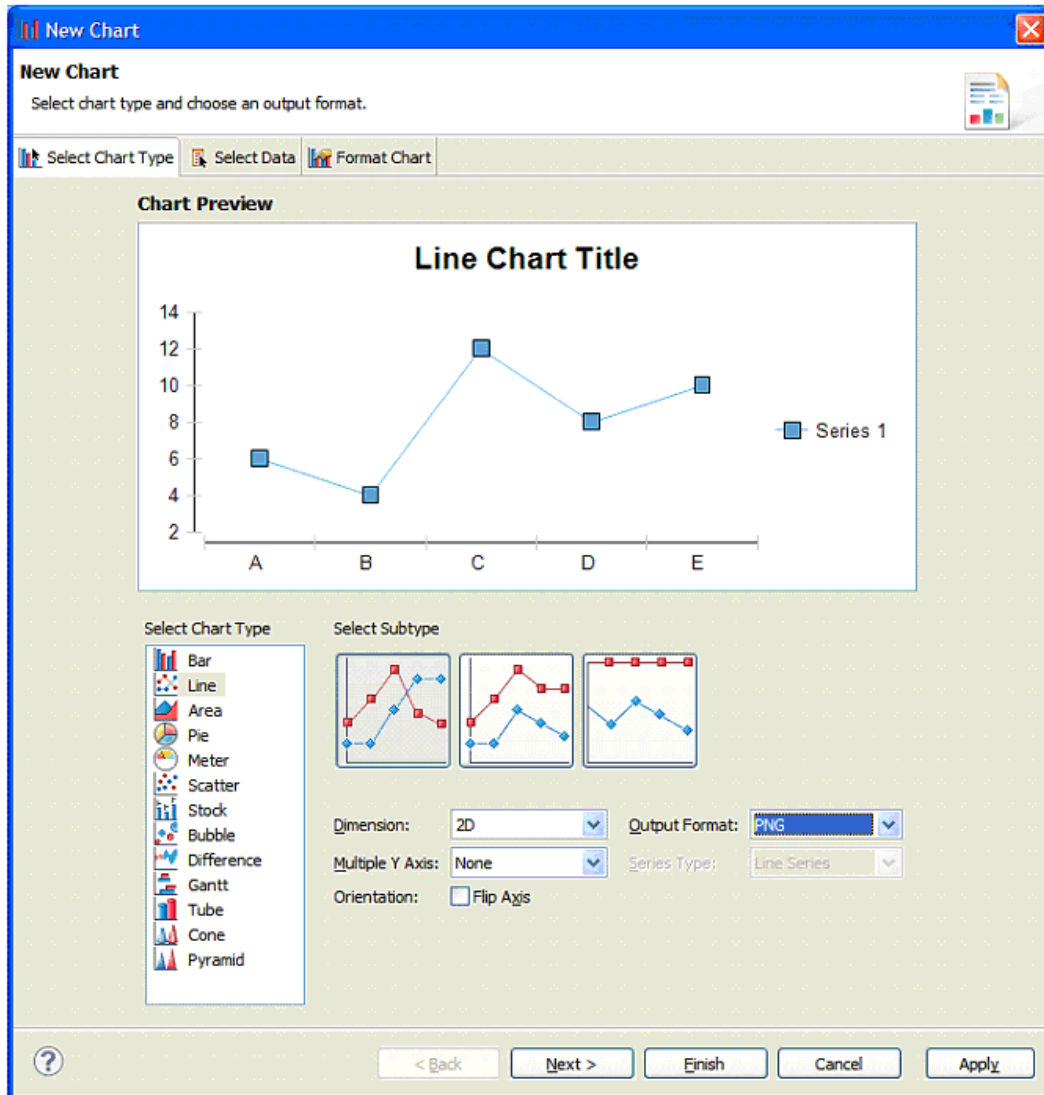


Double-click Data Sources > `ecextract_data` to ensure the XML Data Source Connection refers to the data file you downloaded earlier. In this example the file is `C:\ecextract.xml`.



From the Palette window, drag a Chart object into your report. Your view will be similar to the following:

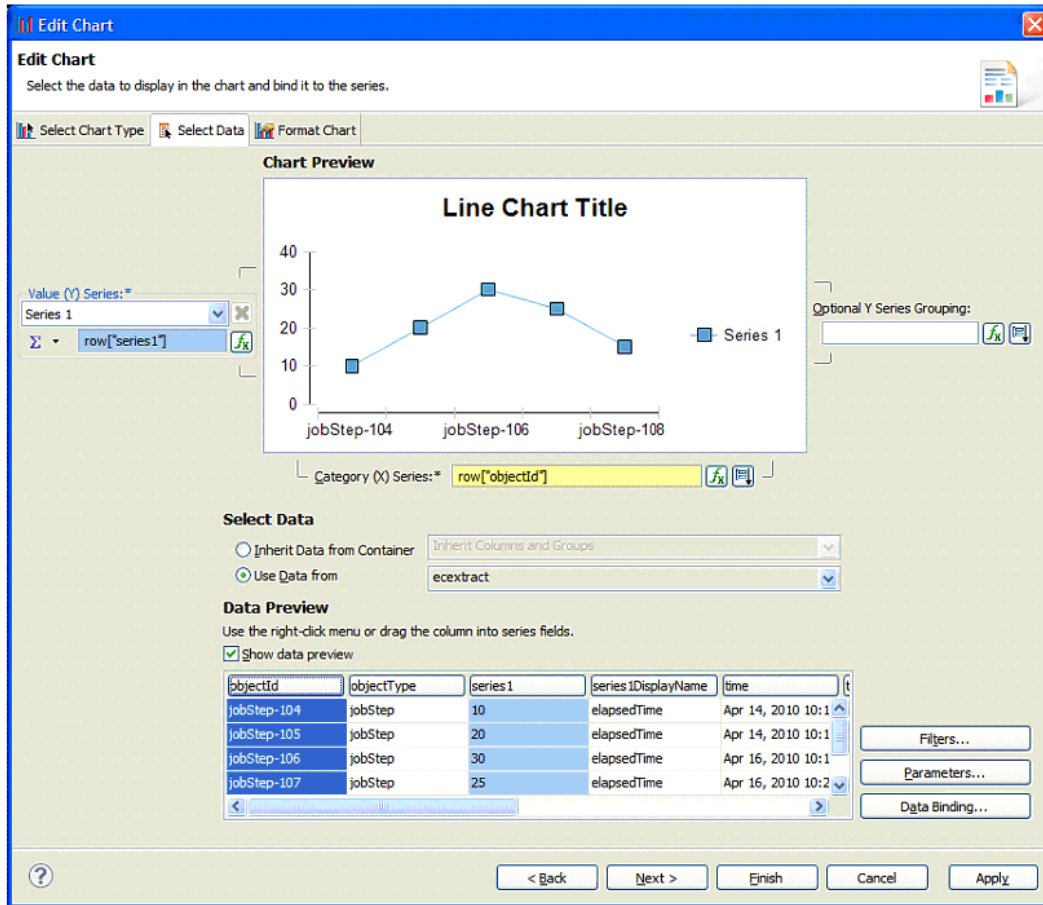
In the New Chart dialog, select Line under Select Chart Type and browse to select PNG for the Output Format.



Click **Next**.

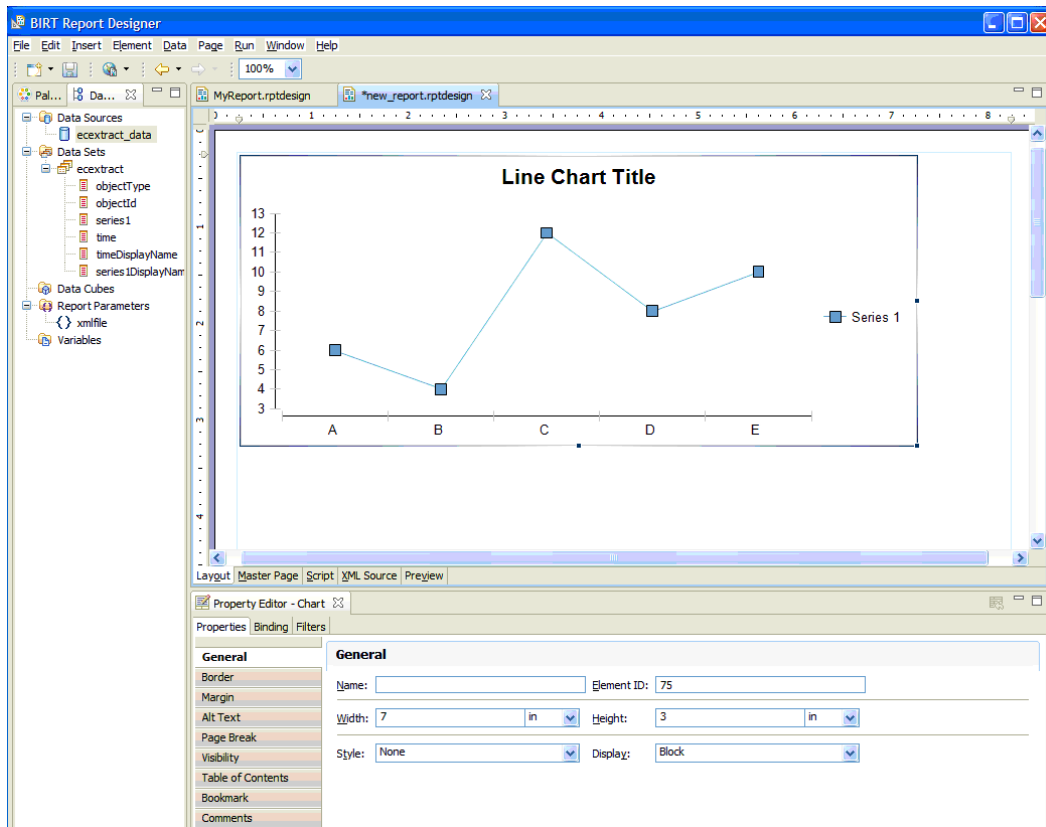
On the next screen:

- Under Select Data, browse to select ecextract.
- Make sure the Use Data From radio button is selected.
- Drag the "series1" header to the Value (Y) Series field.
- Drag the "objectId" header to the Category (X) Series field.



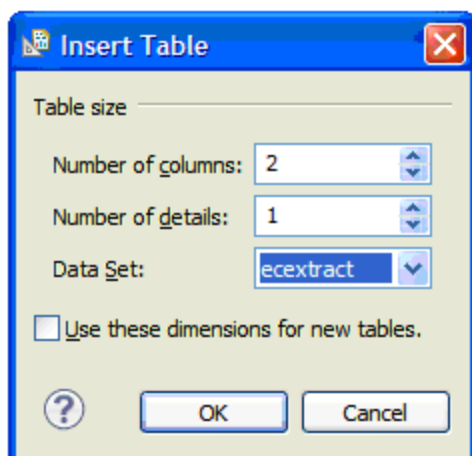
Click **Next** to apply any other formatting changes you choose, then click **Finish**.

On the next screen, after clicking Finish, you have to option to resize the chart as you wish.

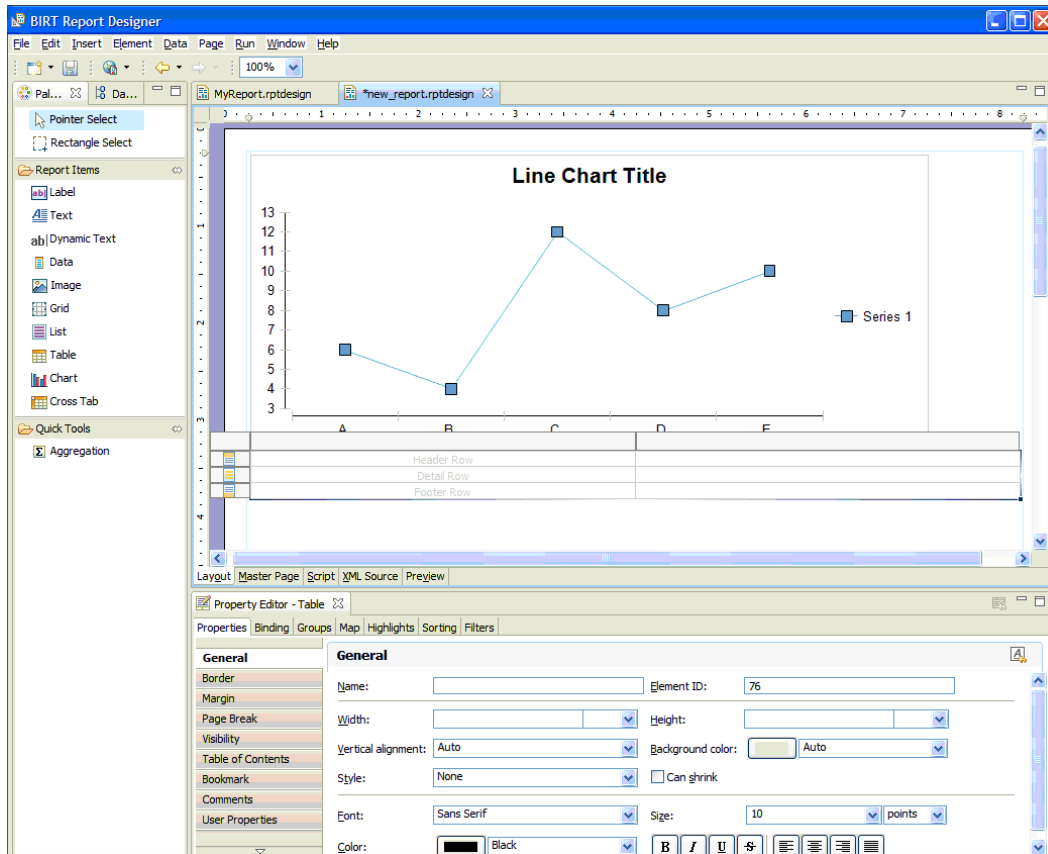


You are now ready to drag a Table item from the Palette to the report layout below the chart.

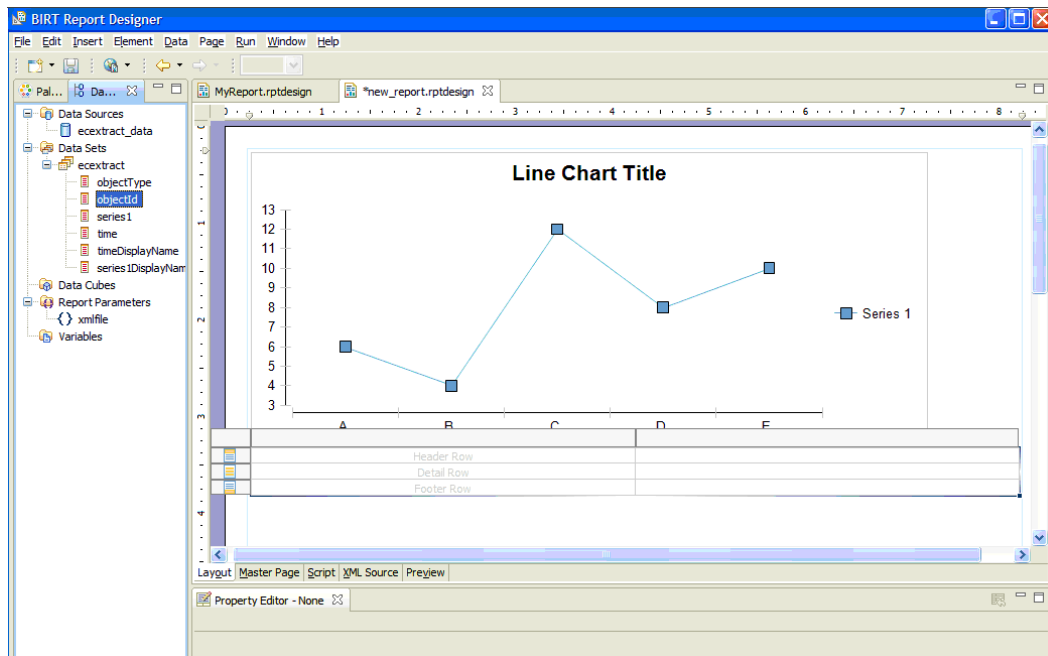
Select "2" columns and the ecextract dataset.



Click **OK**.



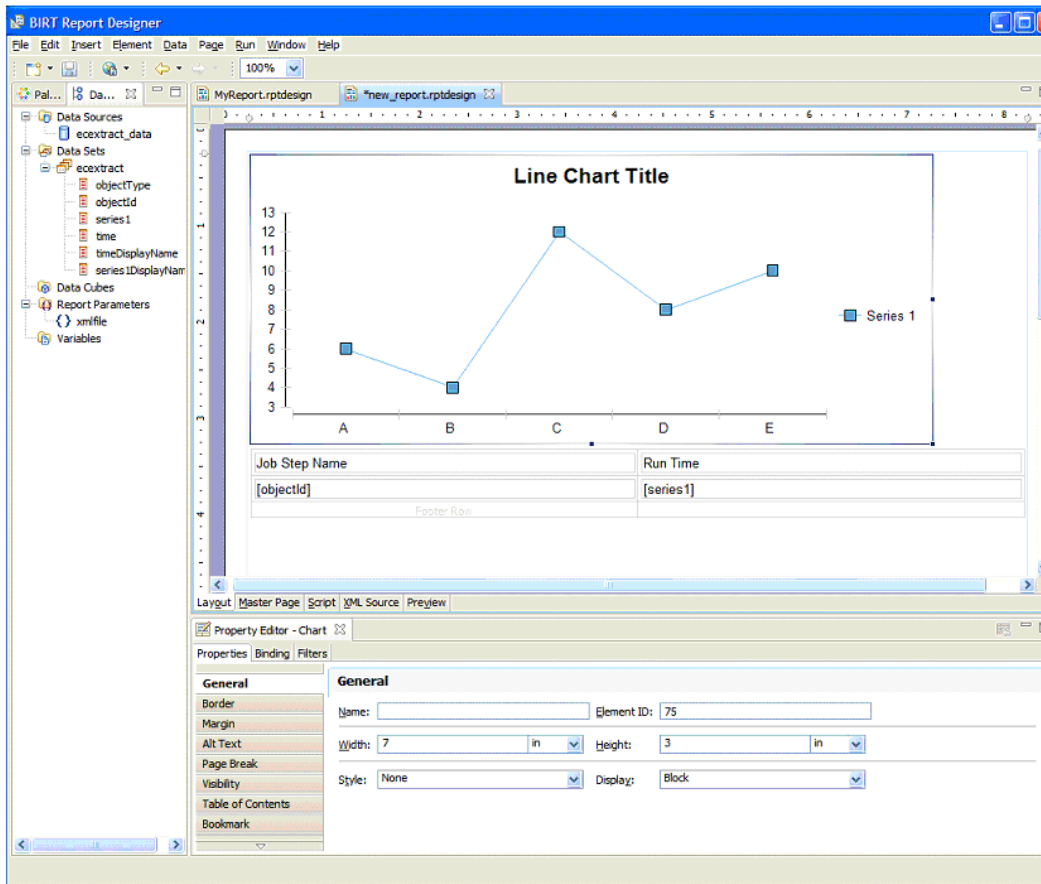
Select the Data Explorer tab and expand Data Sets > ecextract. See the next screen.



Now you are ready to:

- Drag "objectId" to the left Detail Row.
- Drag "series1" to the right Detail Row.
- Replace the Header Row for "objectId" and "series1" with Job Step Name and Run Time, respectively.

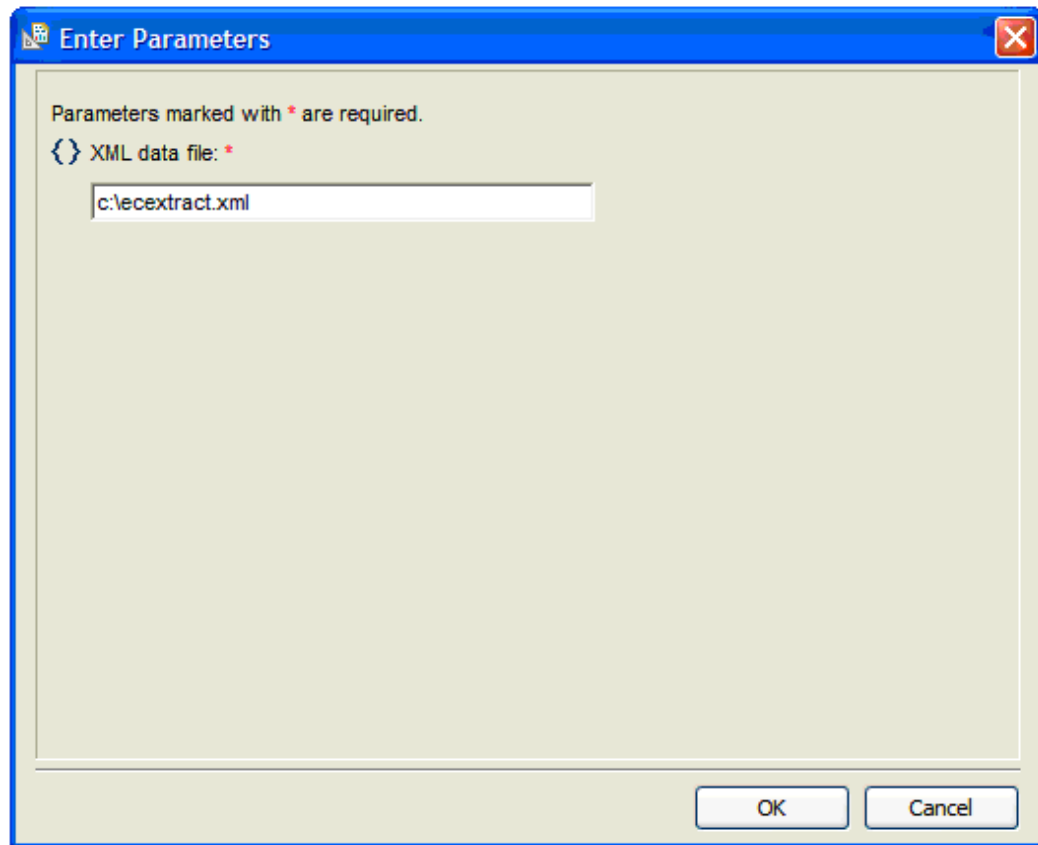
See the next screen example.

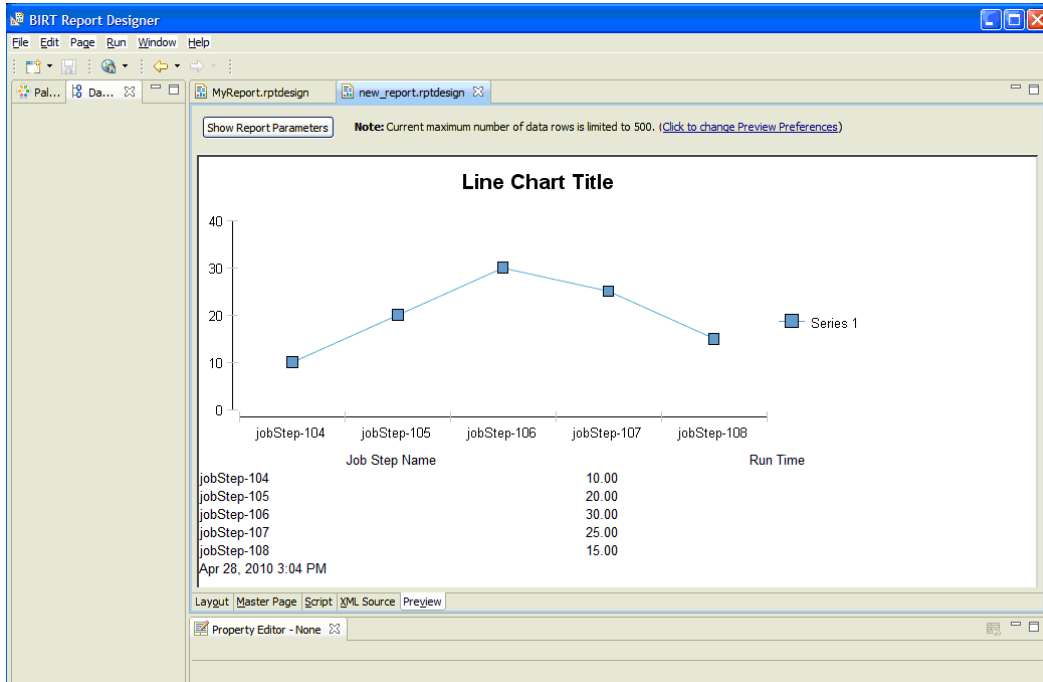


Select Preview to see your report.

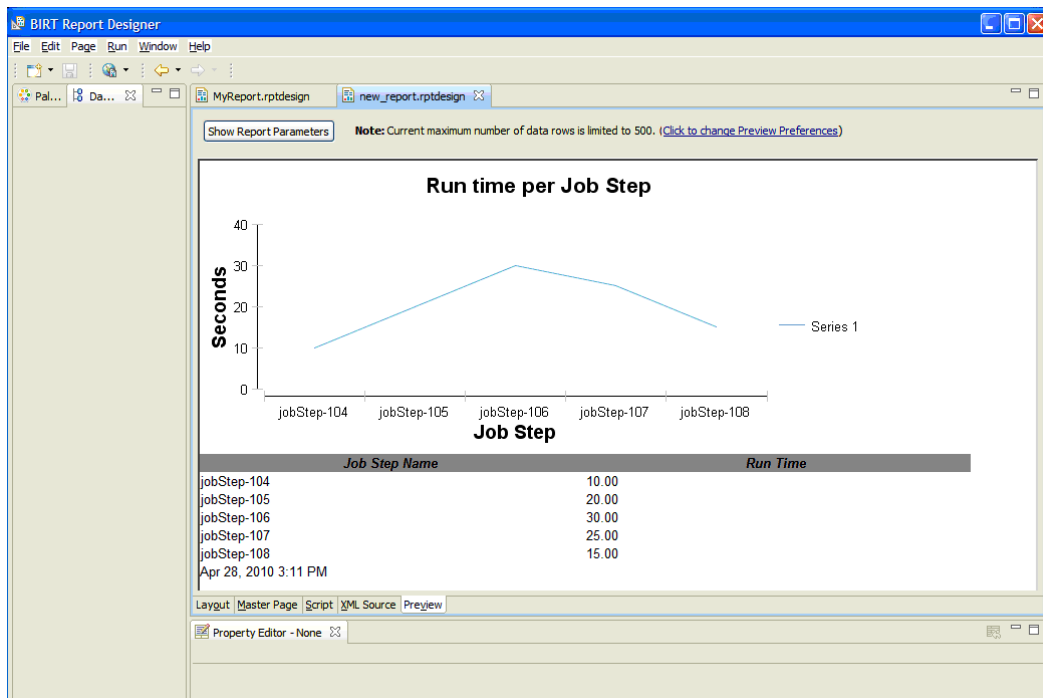
In the "Enter Parameters" dialog, type the location of your ecextract.xml data file.

Click **OK**.





At this point, you can continue to make formatting changes to make your custom report look exactly how you want it to look. Our example report, when finished, looked like the following example::



Deploy your custom report

If you have not already done so, save your new report design now. We have named our report design "JobStepRunTime.rptdesign".

The new report design needs to be stored in a location accessible to the ElectricFlow Agent that will run the report. As in the previous example, we will use the ElectricFlow plugins directory for this purpose.

- Save the JobStepRunTime.rptdesign file to a new JobStepRunTimeReport directory in the ElectricFlow plugins directory.

The default location of the ElectricFlow plugins directory is:

On Linux: /opt/electriccloud/electriccommander/plugins

On Windows: C:\Documents and Settings\All Users\Application Data\Electric Cloud\ElectricCommander\plugins

Linux example - The following commands will make a copy of the Value Over Time report file components and then rename the files:

```
$ cd /opt/electriccloud/electriccommander/plugins
$ mkdir JobStepRunTimeReport
```

- Now copy the previously saved JobStepRunTime.rptdesign file to the JobStepRunTimeReport directory.
- Make a copy of the RunReport_ValueOverTime procedure from the EC-Reports plugin that performs the steps to generate a report, including extracting data, running the report design, and registering the report output to appear on the Report tab of the Project Details page. In this example, we will create a Project called "JobStepRunTime" and the procedure we copy will be called "RunReport_JobStepRunTime".

```
$ ectool createProject JobStepRunTime

$ ectool clone --cloneName /projects/JobStepRunTime/procedures/RunReport_
JobStepRunTime

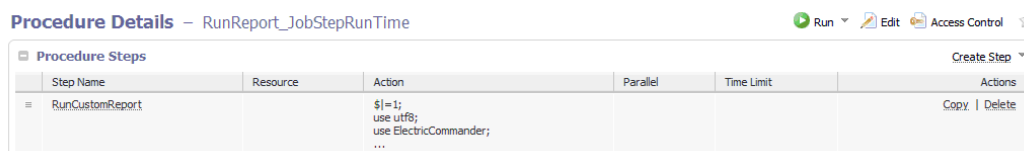
--projectName /plugins/EC-Reports/project --procedureName RunReport_
ValueOverTime

<response requestId="1"

  <cloneName>RunReport_JobStepRunTime</cloneName>

</response>
```

- Edit the "RunReport_JobStepRunTime" procedure in project "JobStepRunTime" to reference the report design file copied earlier.
- Navigate to project JobStepRunTime, procedure RunReport_JobStepRunTime and select the RunCustomReport step to edit the Command block:



Find the following line:

```
our $reportDesignFile =
"$pluginDir/agent/reports/ValueOverTime/ValueOverTime.rptdesign";
```

and change it to:

```
our $reportDesignFile = "$ENV{'COMMANDER_PLUGINS'}/
JobStepRunTime/JobStepRunTime.rptdesign";
```

Edit Step – RunCustomReport

General	
Name:	RunCustomReport
Description:	
Command	
Command(s):	<pre>our \$reportType = "[%Report Type]"; if (\$reportType eq "") { \$reportType = \$reportTitle; } our \$reportFormat = "html"; #html, pdf, doc, xls our \$reportName = safeFileName("\$reportTitle.\$reportFormat"); our \$artifactsDir = "artifacts"; our \$agentInstallDir = getAgentInstallDir(); our \$pluginDir = "\$ENV{'COMMANDER_PLUGINS'}/[%plugins/EC-Reports/pluginName]"; our \$reportDesignFile = "\$ENV{'COMMANDER_PLUGINS'}/JobStepRunTime/JobStepRunTime.rptdesign"; our \$chartInstallDir = "\$ENV{'COMMANDER_PLUGINS'}/[%plugins/EC-ReportEngine]/agent"; our \$ecextractXmlFile = safeFileName("ecextract-\$reportType-[%increment /myJob/xmlCounter].xml"); our \$locales = getLocales();</pre>

Click **OK** to save the change.

Test your new report

Run the report by running the RunReport_JobStepRunTime procedure in the JobStepRunTime project. On the Run Procedure page, change the values of the following parameters:

- Object Type => Job Step
- Property Expression => /1000
- Report Title => Job Step Run Time

Run Procedure – RunReport_JobStepRunTime

Parameters

Chart Time Grouping:

Chart Type:

Chart X-Axis Label:

Chart Y-Axis Label:

Credential: Username: Password:

Locales:

Object Type:

Property:

Property Expression:

Property Function:

Report Title:

Report Type:

Resource:

Saved Filter Project: ☒ Current ☐ Browse

Saved Filter Name: Browse

Table Column Heading Property:

Table Column Heading Time:

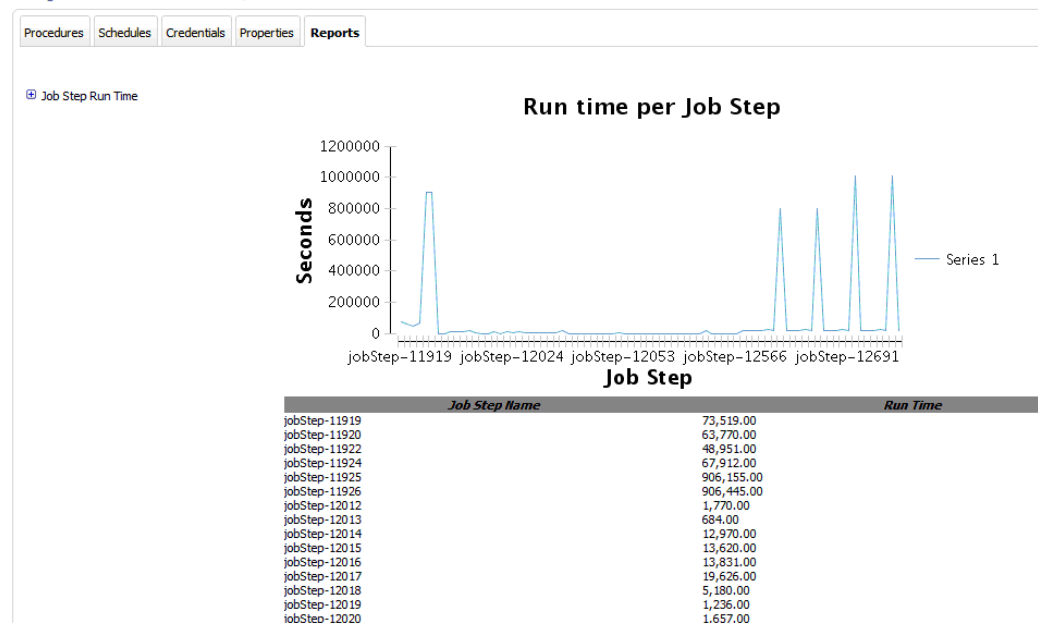
Table Time Grouping:

Time Period:

Click **Run**.

When the job completes, your new report will be available in the Links section of the Job Details page and from the Reports tab in the JobStepRunTime project.

Project Details – JobStepRunTime



System Health Monitoring

You can also use Stasd and Graphite to generate custom reports for system health monitoring. Using Stasd and Graphite is beyond the scope of this Help topic. You can find documentation and tutorials at these sites:

- <https://github.com/etsy/statsd/>
- <http://graphite.readthedocs.org/en/latest/index.html#>
- <https://www.digitalocean.com/community/tutorials/an-introduction-to-tracking-statistics-with-graphite-statsd-and-collectd>
- <https://www.digitalocean.com/community/tutorials/installing-and-configuring-graphite-and-statsd-on-an-ubuntu-12-04-vps>

Workflow Overview

[Workflow objects](#)

[Access Control](#)

[Property Search Paths](#)

[Parameters](#)

[Sending Notifications](#)

[Workflow logs](#)

[A Workflow Tutorial](#)

Use a Workflow to design and manage processes at a higher level than individual jobs. Workflows allow you to combine procedures into processes to create build-test-deploy lifecycles (for example). A workflow contains *states* and *transitions* you define to provide complete control over your workflow process. The ElectricFlow Workflow feature allows you to define an unlimited range of large or small lifecycle combinations to meet your needs.

Some benefits of defining a workflow include...

- **Manual intervention** - workflows make it easy to introduce manual decision points into your process. With workflows, you can enable specific users or groups to choose how the workflow should proceed. Using an email notification, you can notify one or more users about the workflow status. If the workflow is waiting for someone to manually intervene, an email (including a screen link) can provide immediate attention and access to choices for how the workflow should proceed.
- **Aggregation** - workflows allow you to collect information from a set of related jobs into a single location. A workflow can act as a central place to monitor the full lifecycle of your build, test, and deploy process.
- **Looping** - workflows provide the native ability to run a job more than once. You have full control over the loop condition—you can re-run the job any number of times, only re-run the job if it did not succeed, or make this decision based on a custom job property.

- **Branching** - workflows allow you to run multiple jobs in parallel and wait for them to complete before moving to the next state in the workflow. If you combine branching with looping, you can re-run these jobs based on a condition you defined (for example, only re-run jobs that failed).
- **Multipathed** - design a workflow to take multiple paths automatically, depending on the outcome of each state's action, and the pre- and post-triggers you defined for each state.

Basic workflow concepts

Workflows contain states—one or more states can be designated as "starting" states to provide multiple entry points into the workflow. When a workflow is launched, a starting state is specified. You may prefer to design a complex or multi-purpose workflow so you can start the workflow at a different state to complete only a portion of the workflow for a particular outcome. For example, you might choose to run only the test suite workflow section.

As each state in a workflow becomes active, it performs an *action*. The state's action can be to create a job from a procedure or to start a workflow from a workflow definition. When the job or called workflow completes, "On Completion" transitions are evaluated to see if the workflow should change to a different state, possibly based on the outcome of the action.

You can create as many state definitions as you need. For example, you might have a small workflow definition with only 5 states, calling 5 procedures, or you might create 50-100 or more states to process a like-number of procedures from multiple projects.

A state can send email notifications before or after an action to notify interested users about workflow progress or to request manual user intervention to move the workflow to the next state.

Transitions are used to move workflow progress from one state to another state. Four types of transitions are available to move a workflow to the next state:

On Enter – evaluates before sending email notifiers or starting the action

On Start – evaluates immediately after starting the action. These transitions are ignored if no action is specified for the source state.

On Completion – evaluates when the action completes. These transitions are ignored if no action is specified for the source state.

Note: On Completion transitions are taken only if the state is still active when the action completes. These transitions are ignored if the workflow has transitioned to another state.

Manual – evaluates when a user selects the transition in the UI and specifies parameters. The same action can occur using ectool or the Perl API by calling `transitionWorkflow`. Only users who have "execute" permission on the transition are allowed to use the manual transition.

Similar to other objects in the ElectricFlow system, workflow objects can contain access control privileges, properties, and parameters.

Workflow objects

Workflow objects are split into two types: **Definition** objects and **Instance** objects. Definition objects provide the template for a running workflow instance. You create a new workflow by defining a Workflow Definition along with its State Definition and Transition Definition objects.

When you run the workflow definition, the system creates a new Workflow object with an equivalent set of State and Transition objects that represent the run-time instances of the workflow definition.

Note: We omit the "Instance" qualifier for brevity in the API and the UI.

Workflow Definition

This is the top-level workflow object, which is a container for states, and transitions, and other information defining your workflow. In the same way a procedure defines the behavior of a job, the workflow definition defines the behavior of a workflow.

Create a project to get started:

- Select the Projects tab, then click the **Create Project** link. After creating a project, its name will appear in the table on the Projects page.
- Select your project name to go to its Project Details page, select the Workflow Definitions subtab, then select the **Create Workflow Definition** link.
- On the New Workflow Definition page, name your workflow definition. Access the **Help** link on that page if you need help creating the Workflow Name Template (similar to the Job Name Template). Click **Save**.
- Now, on the Workflow Definition Details page (Graph view), you are ready to create the first state for your workflow.

Click [here](#) to see the Help topic for the Workflow Definition Details page that contains an example of how this page might look as you add states and transitions to your workflow definition. Notice also that you can add properties to the workflow definition at this point.

State Definition

Each workflow can contain one or more states. Defining states for a workflow is analogous to defining steps for a procedure.

You might choose to name your states for their intended purpose (for example: waiting for user, building, testing, or test1, test2, and so on) just as you might name steps for your procedure. You can define multiple states as "startable" so you can choose different starting points when you begin using the workflow. By default, the first defined state is always "startable".

As you create states, they will be displayed in the graph on the Workflow Definition Details page.

Selecting a state in the Graph view opens the State Definition panel to define:

- the action you want this state to perform, along with any parameters required by the action,
- transitions for this state,
- parameters for this state,
- email notifiers for this state,
- and any properties you want to assign to this state.

Transition Definition

Each state can contain one or more transitions. The transition definition requires a name for the transition. This transition name will appear on the Workflow Definition Details graph and the Workflow Definition Details List

view.

You can define one or more transitions for each state.

When defining a transition, you can:

- Add a text description for your reference,
- Select the "target" state,
- Select the type of transition trigger you want to use (On Enter, On Start, On Completion, or Manual)
See the [Workflow Definition Details](#) Help topic for trigger type definitions and other information for defining a transition,
- Select (or enter) a Condition to specify when the transition is allowed to trigger,
- and assign parameters to a transition. These parameters are passed to the target state.

Workflow

While we generically refer to or think of a "workflow" as an automated process to perform multiple tasks, it is actually the end result of running a Workflow Definition. Running a Workflow Definition produces a Workflow, which is analogous to running a Procedure, which produces a Job.

After you start running Workflow Definitions, select the main Workflow tab to see all workflows that ran (or are still running) listed in a table, each linked to its own Workflow Details page. The Workflow Details page main view is a graphical representation of the workflow, and also provides links to stop the workflow process, run the workflow again, see the workflow log, and so on.

A Workflow contains a pointer to:

- its workflow definition,
- sets of states and their transitions,
- the current and initial state,
- parameters to the initial state,
- a "complete" value to specify whether or not the workflow is complete,
- and a log containing transitions taken and user events.

State

The run-time instance of a State Definition is a State. The state is part of a workflow and contains information about actual parameter values specified by the user when the workflow was run or from transitions taken to enter the state.

If the state has an *action*, then the state contains information about the most recently created job or workflow. Because a state may be entered multiple times, the state contains a copy of the state definition properties needed to launch the action again.

Note: After the state is created, it no longer depends on the state definition, so any changes made to the definition will not affect workflows already running. Changes do not take effect until the next time the workflow runs.

Transition

The run-time instance of a Transition Definition is a Transition. The transition is part of a state and contains a copy of information from the transition definition. Each transition object corresponds to a transition definition, but once defined, each transition is recognized by its unique name. A transition contains unexpanded actual parameters cloned from definition, expanded and passed to the target state on entry, the Javascript condition, and the trigger type instructing the transition when to occur.

Note: After the transition is created, it no longer depends on the transition definition, so any changes made to the definition will not affect workflows already running. Changes do not take effect until the next time the workflow runs.

A transition moves the workflow from its state to a *target* state.

Access Control

Setting access control privileges for those persons who can or cannot run a workflow or execute a manual transition may relate to security policies in your organization.

Specific workflow access control notes:

- All workflow, state, and transition ACLs are copied from the definition object to their corresponding instance when the workflow is run initially.
- To run a workflow, you must have execute permission on the state definition you are using to start the workflow.
- To take a manual transition, you need execute permission on that transition.
- To strictly control who can take a manual transition:
Grant read and modify privileges to anyone who can view or edit the workflow, but only give execute permission to those users or groups who can take the manual transition.
- You can control who can start each of the startable states by setting ACLs on state definitions accordingly.

For more information about access control, including examples you might use, see the [Access Control](#) Help topic.

Property Search Paths

A workflow serves as a hub of information that can be shared between states, transitions, jobs, and other workflows.

Depending on the context, an implicit search path is used to reference a property by name. Using a simple property name like `$(someCustomProperty)` “walks” down the following paths until it finds a property by that name. If the search fails to find a property, an error is produced.

The search path followed to resolve a simple property name depends on where the reference occurs:

Transition context:

1. Transition property sheet
2. State property sheet – includes parameter values passed to that state and any custom properties. Because a state can be entered multiple times, parameter values are available for the most recent entry only.
3. Starting state property sheet – used to get to parameter values for the initial entry into the workflow.
4. Workflow property sheet – used to access information shared across the workflow.

State context:

1. State property sheet – includes parameter values passed to that state and any custom properties. Because a state can be entered multiple times, parameter values are available only for the most recent entry.
2. Starting state property sheet – used to get to parameter values for the initial entry into the workflow.
3. Workflow property sheet – used to access information shared across the workflow.

For information on context-relative property path shortcuts that allow passing information from one object to another or accessing a property from different parts of the workflow, review the [Properties](#) Help topic. The Properties Help topic also includes all intrinsic workflow-related properties.

Parameters

States may use formal parameters to determine what information to provide when the workflow enters that state. Parameter values can come from the initial run workflow call or from a subsequent transition into the state.

An automatic transition must specify all values required by the target state. A manual transition may omit some values to be passed to its target state. If a manual transition does not specify all values needed by the target state, then any missing values are collected from the user when the transition is taken.

Parameters to the starting state for the workflow are collected when the workflow is launched. These parameters are accessible throughout the workflow and act like global workflow parameters. See the [Property Search Paths](#) section above.

It is important to distinguish between formal parameters on a state and formal parameters on the action the state calls (procedure or workflow starting state). The procedure (or workflow) may have its own parameters, so the state is responsible for entering required actual parameter values. In some cases, values may be static values, in other cases the values may come from the state's parameters or those of the starting state (for example, global workflow parameters)

Parameters on the calling state do not need to be the same as the parameters on the action:

- A state parameter value can be passed as the value for an action's parameter by specifying a property reference in the value field on the Action tab for the state definition.
- All parameters on workflow current and starting states are accessible with a simple property reference (for example, '\$[param1]')
- Parameters on a workflow starting state act like global workflow parameters because they are available everywhere within the workflow.

Sending Notifications

A state can be defined to send email notifications when it is entered, when it starts its action, and/or when the action completes. Notifiers are created by visiting the Notifications tab on the State Definition panel. See the [Email Notifier](#) Help topic for details on defining email notifiers. Sample notifier templates are provided also.

Workflow Logs

Each workflow keeps a log of events that relate to the workflow. The log contains information about the evaluation of transition conditions, transitions taken, actions started, and other information that might prove useful when debugging a workflow. Access the log by clicking the **View Log** link on the Workflow Details page. **For more information**, click [here](#) to go to the Workflow Log Help topic, which contains a screen example of the Workflow Log page.

Visualizing a Workflow

Two Workflow web pages, Workflow Definition Details and Workflow Details, open to the Graph view.

- Workflow Definition Details page - As you build your workflow, you can see a visual representation of state and transition definitions you define.
- Workflow Details page - displays a graph, updated in real-time, for your running or completed workflow.

Your workflow Graph (on either page) contains context-sensitive links to various options. Options are different depending on whether or not you are building or running your workflow. Workflow Definition Details and Workflow Details Help topics include information about the context-sensitive link options provided within the workflow graph.

The next section, "Building a Workflow - a Tutorial," begins with a "sketched" representation of the workflow the tutorial will create. At the end of the tutorial, see the actual workflow graph created by the ElectricFlow workflow process.

Building a Workflow - a Tutorial

This overview tutorial is an example of how to create a "build-test-approve" workflow, building each state and transition we want the workflow to include, and building the workflow in the order we need to use to achieve our purpose. The diagram below outlines our workflow process.

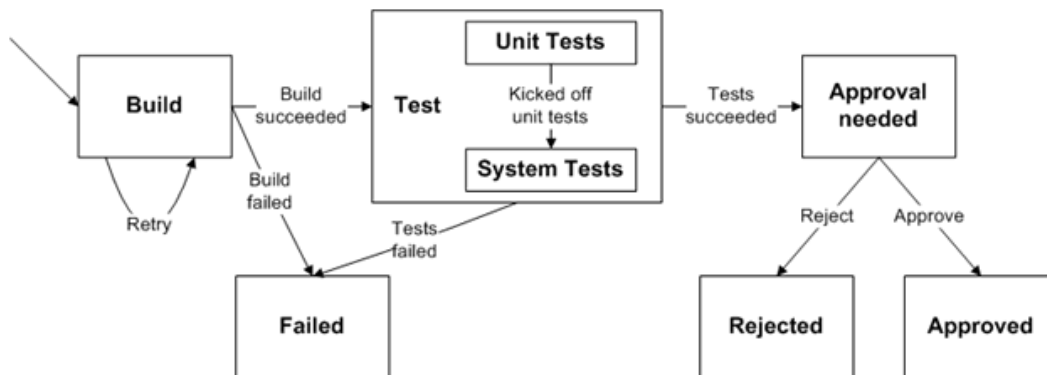
You can read through the tutorial or actually recreate this example workflow on your machine. If you want to use this tutorial as a real, working example, you must do some initial setup first:

- Create a project named "Sample"
- Create a procedure name "BuildProduct"
- On the "BuildProduct" procedure, add a parameter named "branch"

The following diagram is a visual illustration of the workflow we are going to build.

Reading the diagram:

- the starting state is *Build*, which could transition to the *Test* state or the *Failed* state
- two test states - *Unit Tests* and *System Tests*, which could transition to a *Failed* state or an *Approval needed* state
- the *Approval needed* state could transition to one of two states - *Rejected* or *Approved*
- all transitions are represented by the lines between the states, which include "arrows" to show the possible workflow directions (depending on the outcome of each state's action)



Best Practice tip

Designing your workflow on paper or whiteboard, creating your own state and transition diagram, is a great way to organize your thoughts and map your workflow process.

A summary of the component sections to build our workflow:


- [Calling a job from within a workflow](#)
- [Collecting a parameter when a workflow is launched and passing its value to a job](#)
- [Retrying a state](#)
- [Automatically transitioning to different states based on the outcome of a job](#)
- [Invoking another subworkflow](#)
- [Running jobs in parallel](#)
- [Automatically transitioning to different states based on the outcome of jobs in a workflow](#)
- [Waiting for manual intervention](#)
- [Restricting who can take a manual transition](#)
- [Sending email notifiers](#)
- [Adding a global parameter to use later in the workflow](#)

- [Setting the name of your workflow](#)
- [Workflow list](#)

To begin...Calling a job from within a workflow

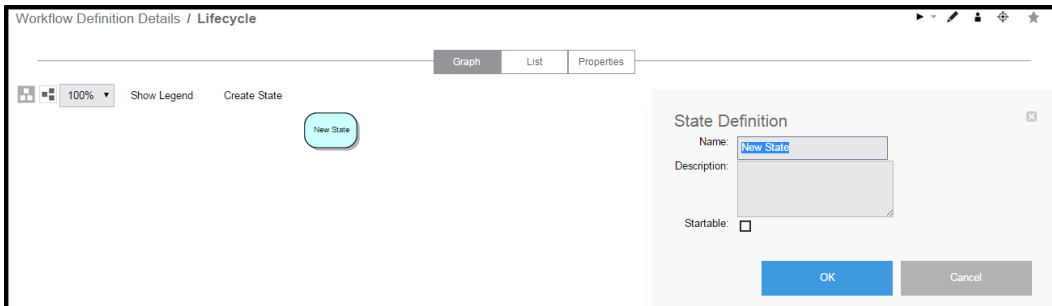
Or, running a job by calling its procedure from a workflow state.

- Select the Projects tab, select the Sample project to go to the Project Details page, then select the Workflow Definitions subtab.
- Click the **Create Workflow Definition** link to go to the New Workflow Definition page..
- Create a new Workflow Definition named *Lifecycle*.



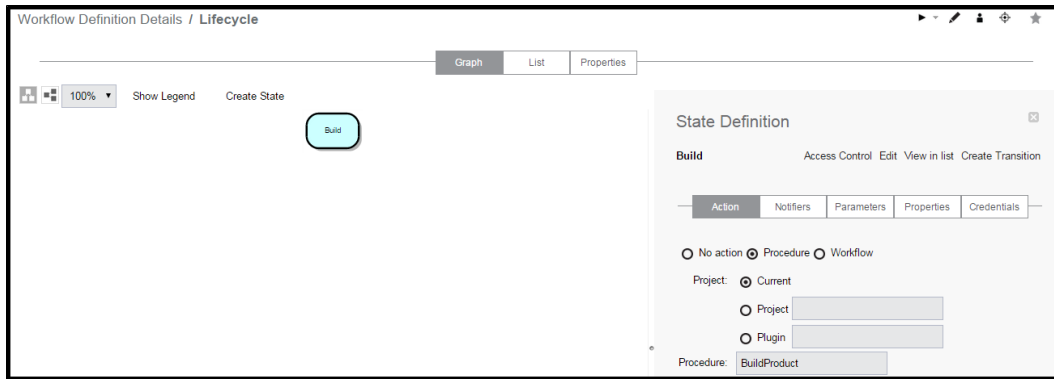
The 'New Workflow Definition' dialog box is shown. It has a title bar with a star icon. Inside, there are three input fields: 'Name' with the value 'Lifecycle', 'Description' (empty), and 'Workflow Name Template' (empty). At the bottom right, there are 'OK' and 'Cancel' buttons.

Because the workflow definition, *Lifecycle*, is a new definition and does not have any state or transition definitions, the State Definition panel opens, ready for you to create a new state definition for your workflow.



The 'Workflow Definition Details / Lifecycle' panel is shown. It has a title bar with a star icon. Below the title bar, there are tabs for 'Graph', 'List', and 'Properties'. The 'Graph' tab is selected. On the left, there is a '100%' zoom dropdown, a 'Show Legend' button, and a 'Create State' button. A 'New State' button is also visible. On the right, the 'State Definition' panel is open, showing a 'Name' field with the value 'New State', a 'Description' field (empty), and a 'Startable' checkbox (unchecked). At the bottom right, there are 'OK' and 'Cancel' buttons.

- Change the system-generated state Name to *Build* and click **OK**.
- The State Definition panel is displayed.



- Specify the Action for this state, which is Procedure.
In this example, the procedure *BuildProduct* already exists with a single parameter named *branch*.
- Set the value of this parameter to "3.0".
- The *Build* state now has an Action associated with it.
- Click **OK**.
Note the shape of this state in the graph. The state shape will be different depending on the state Action selected.
- Click **Run** to launch the workflow.
- Running the workflow shows a Job was launched by the *Build* state. The workflow job completed successfully, indicated by the "check mark" on the state.
- You can get more details about the job by clicking on the *Build* state, which opens the State Details panel, and clicking on the Job link displays the Job Details page.
- Alternately, clicking on the Show History button opens the History panel to show what ran and the workflow outcome.

Collecting a parameter when a workflow is launched and passing its value to a job

Or, "prompting" for a parameter value when a workflow is launched and passing its value to a procedure.

- Return to the Workflow Definition page for the **Lifecycle** workflow.
- Click the *Build* state (from the graph) to open the State Definition panel, then select the Parameters tab.
- Click the **Create Parameter** link to display the New Parameter page.
- Create a parameter called "branch".
- Add a description and make sure the Required? box is "checked" and click **OK**.
- On the Workflow Definition Details page again...
- Change the value passed to the job from 3.0 to `$(branch)`.

Now, when the workflow is launched, you will be prompted for the value of *branch* and you can specify a different value if necessary.

For example, a "3.5" value is now specified in the screen below.

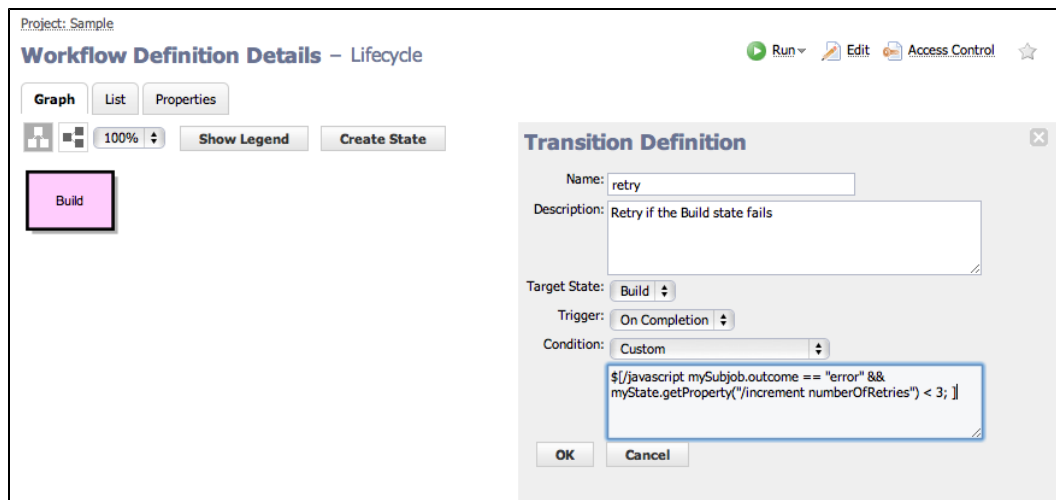
The value is passed to the job when the workflow is launched.

Retrying a state

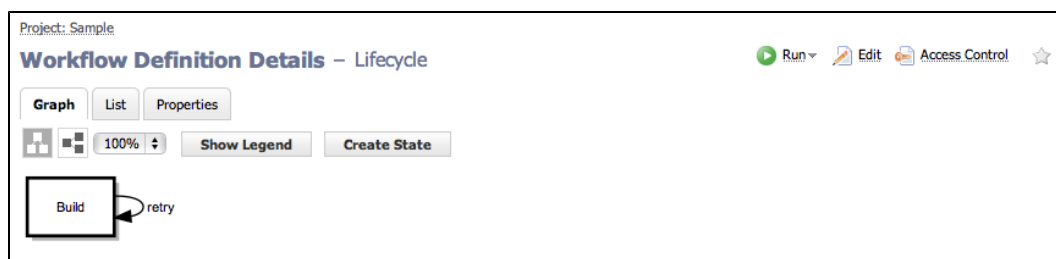
If a State fails (technically, the job started by the workflow state), you may choose to retry it a few times before giving up.

- Create a new transition from the *Build* state back to itself.
- Right-click on the *Build* state and select the **Create Transition** link.
- Create a new On Completion transition named "retry" from the *Build* state that returns to the same state when the job fails.
- Add a limit of 3 retries by using the following JavaScript condition:

```
$/javascript mySubjob.outcome == "error" && myState.getProperty('/increment  
numberOfRetries') < 3; ]
```



- The new transition is displayed on the Workflow Definition Details page.



- When we run the workflow, we see only the last job that was run.

Project: Sample

Workflow Details – workflow_82_201204251214

Run Again Delete View Log Access Control

Final State

Build

Subjob: job_14211_201204251214

General Information

Project: Sample

Workflow Definition: Lifecycle

Graph States Parameters Properties

100% Show Legend Show History

Build retry

- Clicking the States tab and then the **Show All** link displays all jobs that were invoked by the *Build* state.

Project: Sample

Workflow Details – workflow_82_201204251214

Run Again Delete View Log Access Control

Final State

Build

Subjob: job_14211_201204251214

General Information

Project: Sample

Workflow Definition: Lifecycle

Graph **States** Parameters Properties

State Name	Action
Build	job_14211_201204251214 [Show All]

On this screen, you can see the build was attempted 3 times because that was the limit we specified in the JavaScript condition for the "retry" transition.

Job Search Results

3 Results for "callingStateId" equals "282"

New Search Save Filter Edit Search Refresh Search

Job	Status	Priority	Procedure	Launched By	Elapsed Time	Start Time	Actions
job_14211_201204251214	Error	normal	Sample:BuildProduct	workflow_82_201204251214: Build	00:00:00.453	2012-04-25 12:14:18 PDT	Delete
job_14210_201204251214	Error	normal	Sample:BuildProduct	workflow_82_201204251214: Build	00:00:00.613	2012-04-25 12:14:17 PDT	Delete
job_14209_201204251214	Error	normal	Sample:BuildProduct	workflow_82_201204251214: Build	00:00:00.539	2012-04-25 12:14:17 PDT	Delete

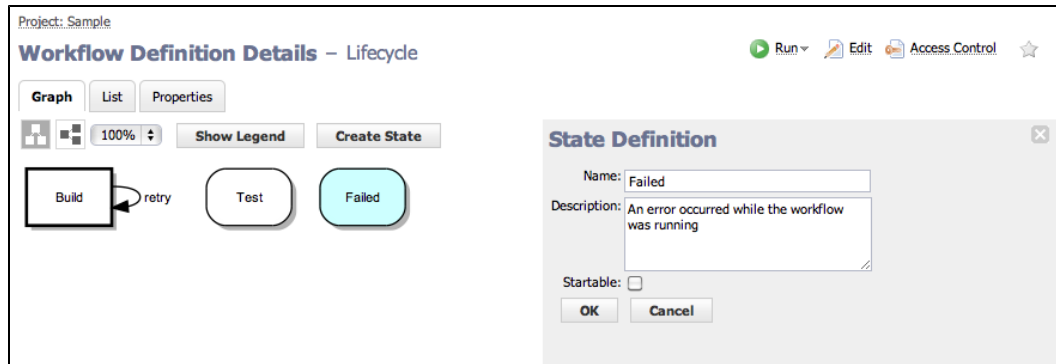
Records per page: 20 1 thru 3 of 3

Also, you can see the number of times a state was visited and which transitions were taken by clicking the **Show History** button.

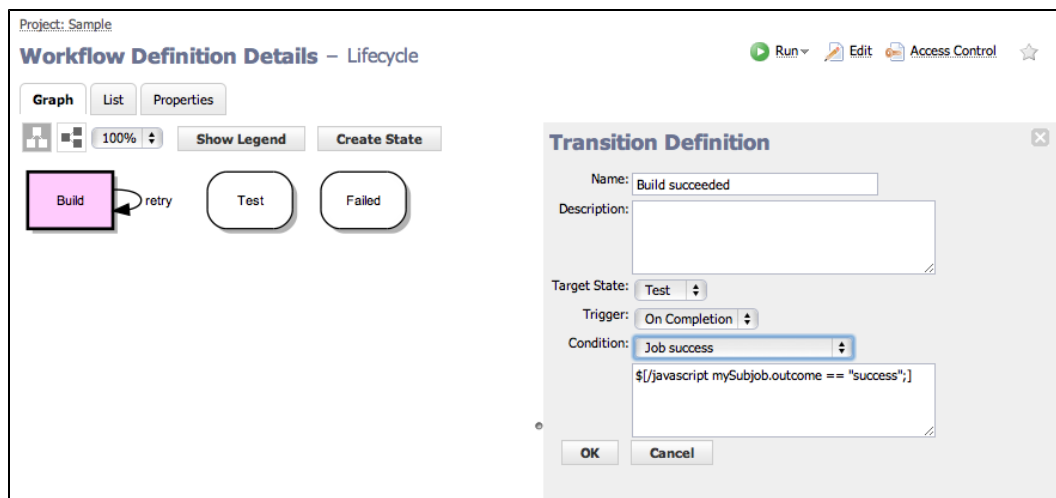
Automatically transitioning to different states based on the outcome of a job

If the *Build* job fails, we might want to end the workflow. If the job succeeds, we might want to continue to the next stages of the workflow.

- Add new states to transition to, depending on the outcome of the job:
 - Create the *Test* state where the workflow will transition to when the build state succeeds.
 - Create a *Failed* state.

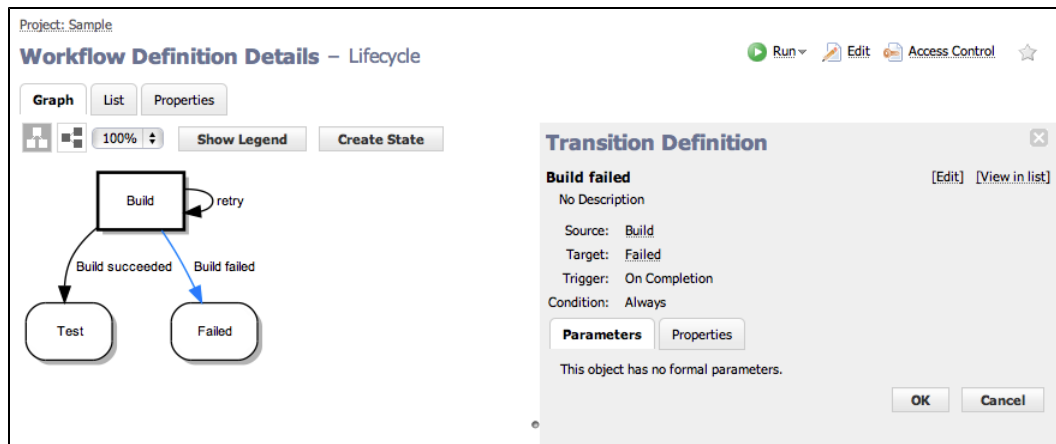


- Create an On Completion transition from *Build* to *Test* when the job succeeds.
- Notice that the Condition dropdown menu provides a template condition called *Job success* that can be used here.

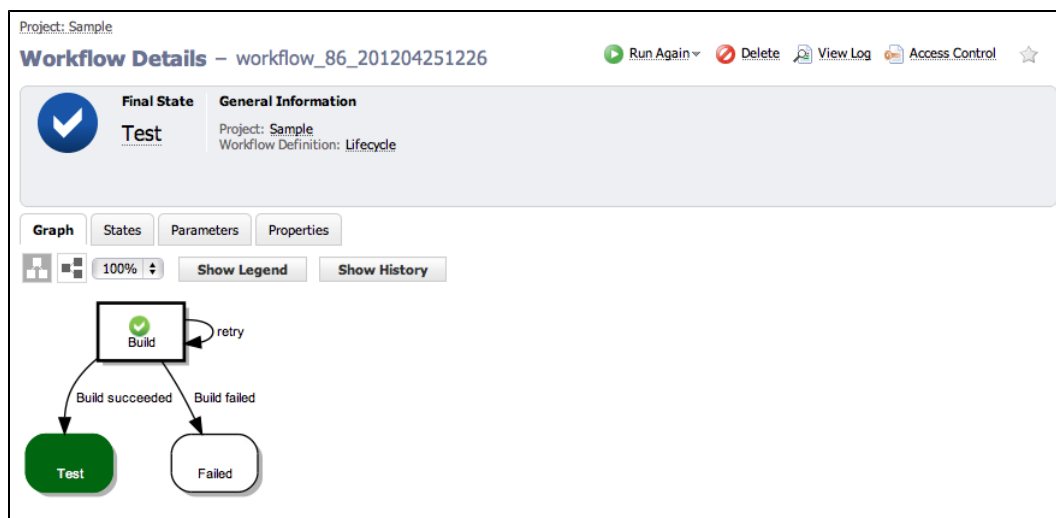


- Create an On Completion transition from *Build* to *Failed* when the job fails.
- Notice that the Condition field remains blank.
Because transitions are evaluated in order, if the workflow gets to this transition, we will know that the "success" case will be false.

Leaving the Condition field "Always" allows us to handle all "non-successful" outcomes (error or warning).



- When you run the workflow, if the build succeeds the workflow will end up in the *Test* state.



- The workflow will end up in the *Failed* state if the build fails after trying three times. (In this example, we show the Workflow History panel open.)

Project: Sample

Workflow Details – workflow_87

Run Again Delete View Log Access Control

Final State **Failed**

General Information
Project: Sample
Workflow Definition: Lifecycle

Graph States Parameters Properties

100% Show Legend Hide History

History

Build	job_14240_201204251227
retry	
Build	job_14241_201204251227
retry	
Build	job_14242_201204251228
Build failed	
Failed	

Invoking another workflow

Just as a state can invoke a job as its “Action”, it can invoke another workflow, which then becomes its subworkflow.

- From the Workflow Definition Details page for *Lifecycle*, click on the *Test* state in the graph.
- Specify the “Action” for the Test state, which is a Workflow.
- In this example, the workflow *AutomaticTests* already exists and so does a state called *UnitTests*, which has a single required parameter called *buildName*.
- We can get the name of the build from the *Build* state by using the property path `$/myWorkflow/states/Build/subjob/jobName`.

Project: Sample

Workflow Definition Details – Lifecycle

Run Edit Access Control

Graph List Properties

100% Show Legend Create State

State Definition

Test [Edit] [View in list] [Create Transition]

Action Parameters Notifiers Properties

☐ No action ☐ Procedure ☒ Workflow

Project: ☒ Current ☐ Project ☐ Plugin

Workflow: Automatic Tests

Starting State: UnitTests

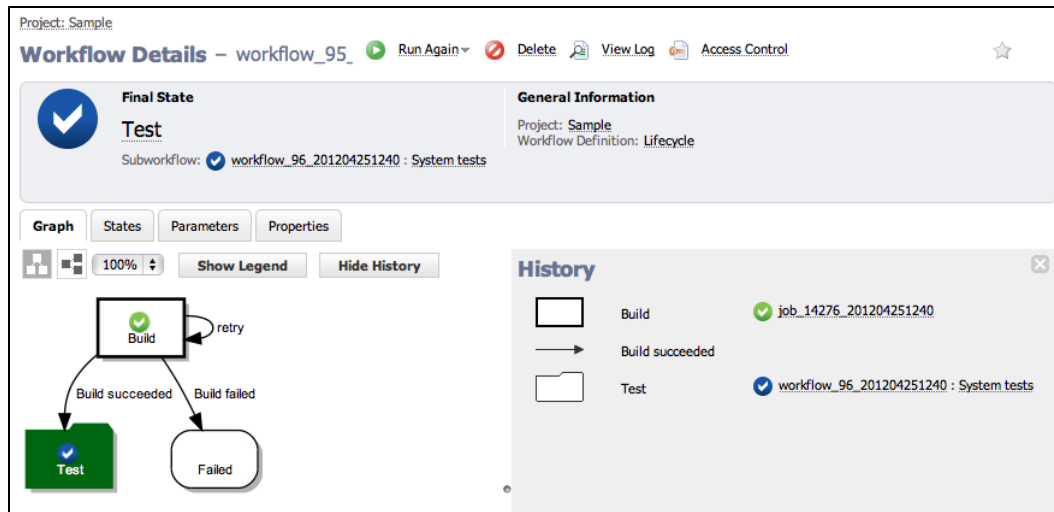
Parameters:

buildName: `$/myWorkflow/states/Build/subjob/jobName`

Saved

OK Cancel

- The *Test* state now has an Action associated with it.
- Because the *Test* state Action is Workflow, the object shape in the graph has changed.

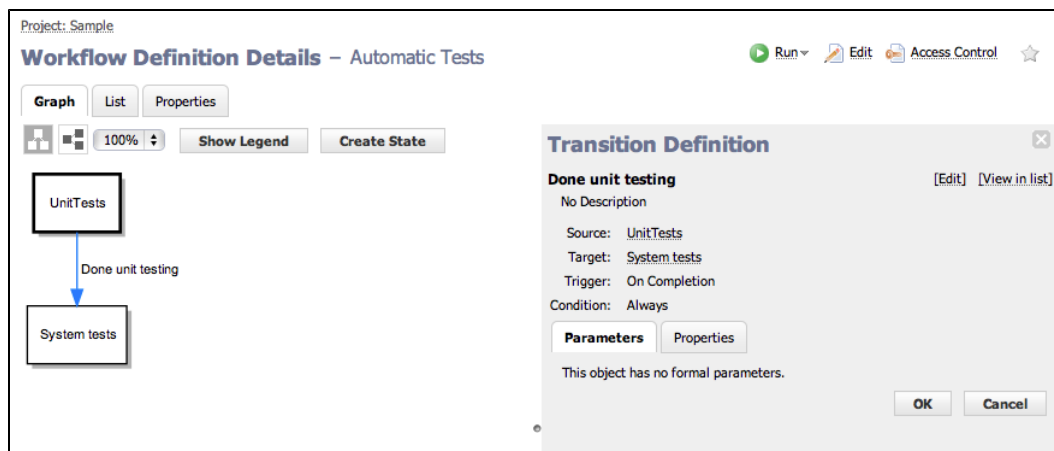


- Running the workflow shows another workflow was launched by the *Test* state.
- Click on the workflow link in the History panel to see more details about that workflow.

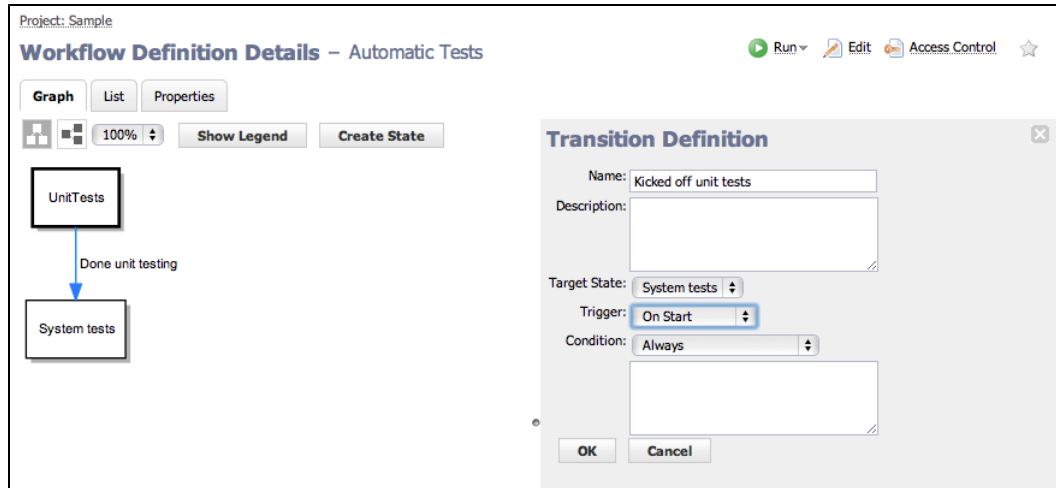
Running jobs in parallel

Run multiple jobs at the same time.

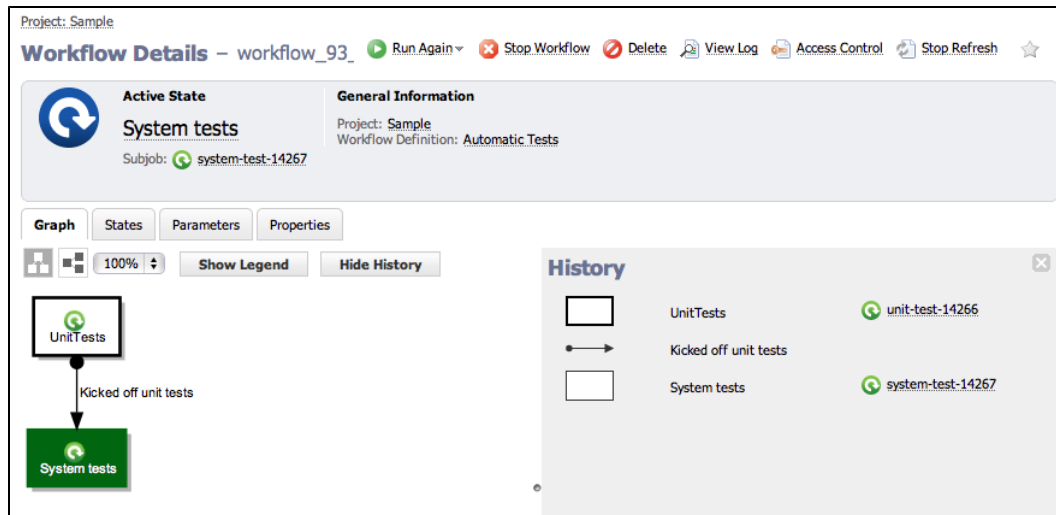
- Look at the *AutomaticTests* workflow that was invoked (in the previous example) by the *Test* state in our main *Lifecycle* workflow. The first state in this workflow runs unit tests, and when the tests complete, the state automatically transitions to the second state that runs system tests.



- Now we will change the behavior of the transition to move from the *Unit tests* state to the *System tests* state as soon as the first job is launched.
- Edit the existing transition by renaming it to *Kicked off unit tests* and changing it to an On Start trigger type.



- Now, when the *AutomaticTests* workflow is launched, both jobs will run in parallel.
- The workflow completes automatically when both jobs complete.



Automatically transitioning to different states based on the outcome of jobs in another workflow

If the jobs in the *Test* workflow fail, we will want the workflow to end. If the jobs succeed, we will want to continue to the next stages of a calling workflow.

- Click the **Create State** link.
- Create the *Approval needed* state that the workflow will transition to when tests succeed.
- Create a transition named "Tests succeeded" from *Test* to *Approval needed* when all jobs in the workflow succeed.
(Right-click on the *Test* state to create the transition.)
- The following JavaScript condition checks the outcome of all subjobs for a subworkflow and evaluates to "true" only if they all succeeded:

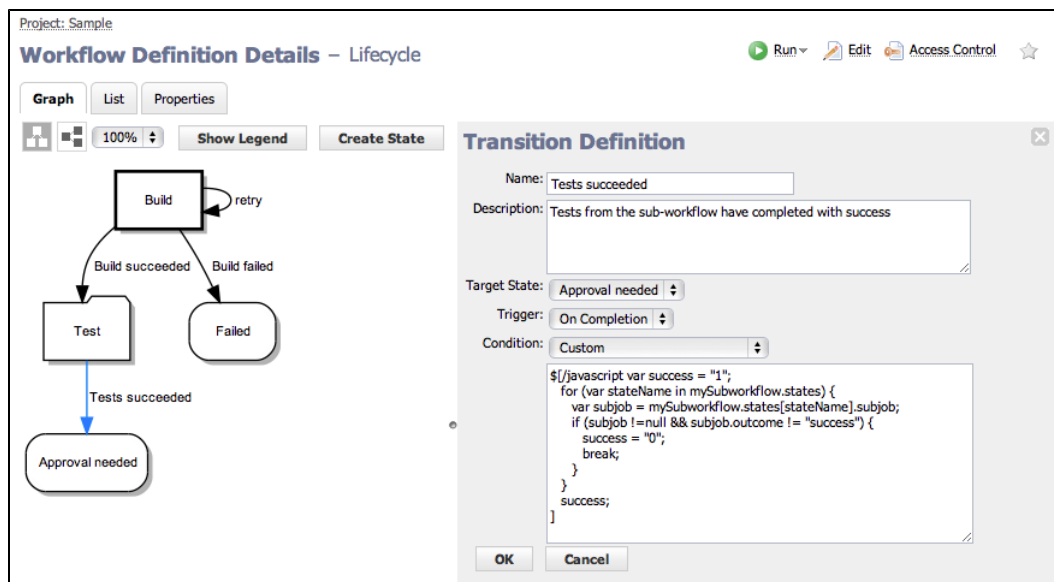
```

$[/javascript
  var success = "1";

  for (var stateName in mySubworkflow.states) {
    var subjob = mySubworkflow.states[stateName].subjob;
    if (subjob !=null && subjob.outcome != "success") {
      success = "0";
      break;
    }
  }

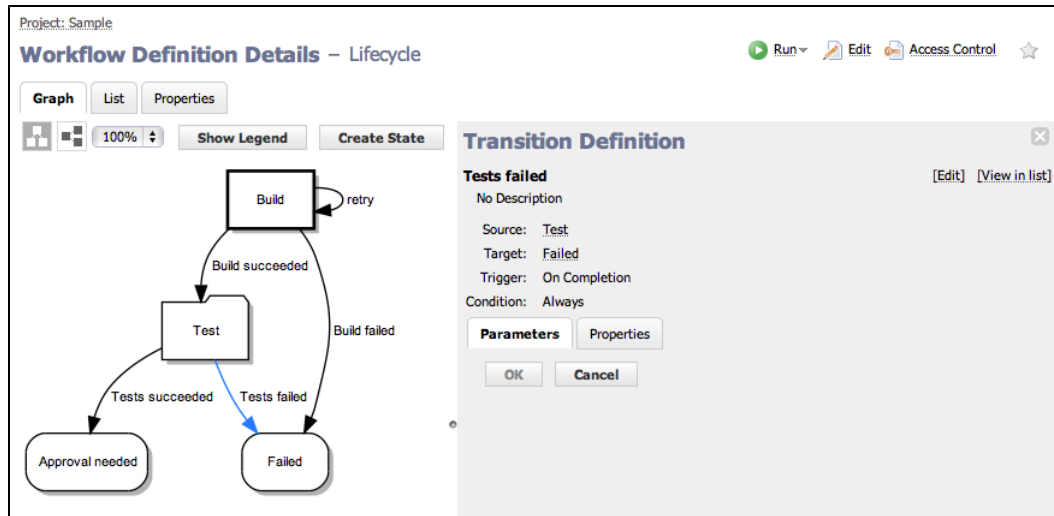
  success;
]

```

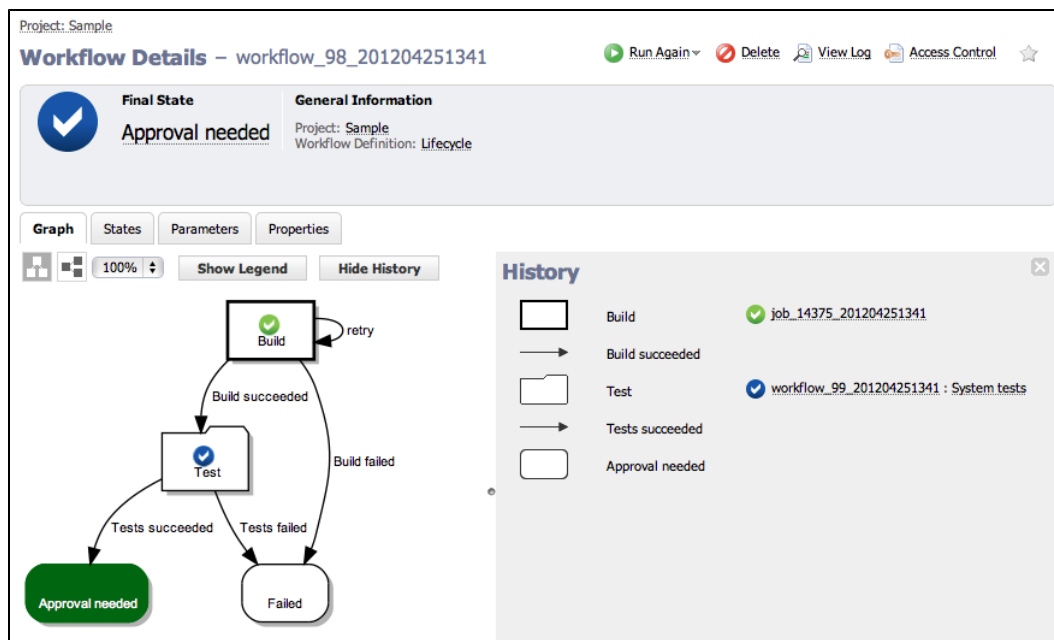


Note: We created the *Approval needed* state first, then we created the *Tests succeeded* transition to connect the two states.

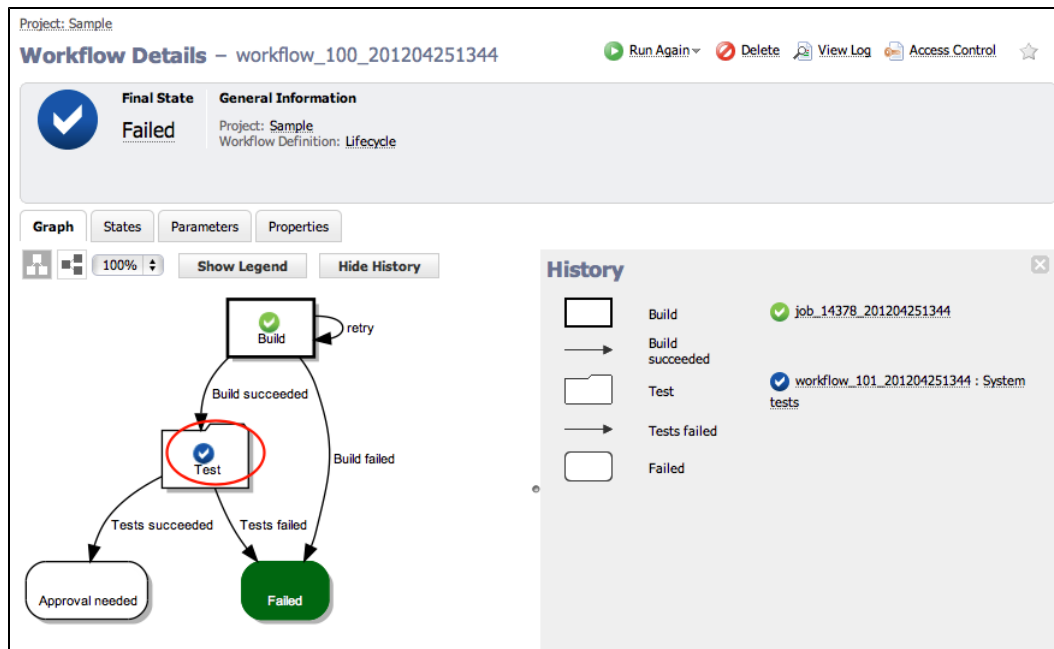
- Create a transition from the *Test* to the *Failed* state when either of the jobs in the workflow fails.
- Notice the Condition field remains blank.
- Because transitions are evaluated in order, if the workflow gets to this transition we know the success case was "false".



- When you run the workflow, if the test jobs succeed, the workflow will end up in the *Approval needed* state.

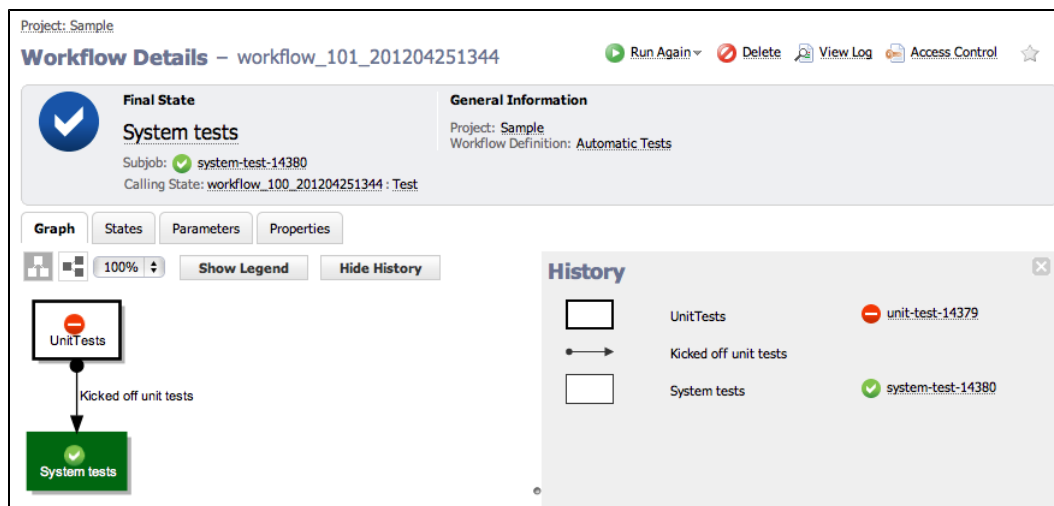


- In the following example, one of the tests in the *Test* workflow encountered an error, so the workflow will end up in the *Failed* state if either of the test jobs fail.



Note: The "blue checkmark" indicates the workflow (or subworkflow) has completed, which means it is no longer running. This icon does NOT indicate the workflow outcome (completed successfully, or with warnings or errors).

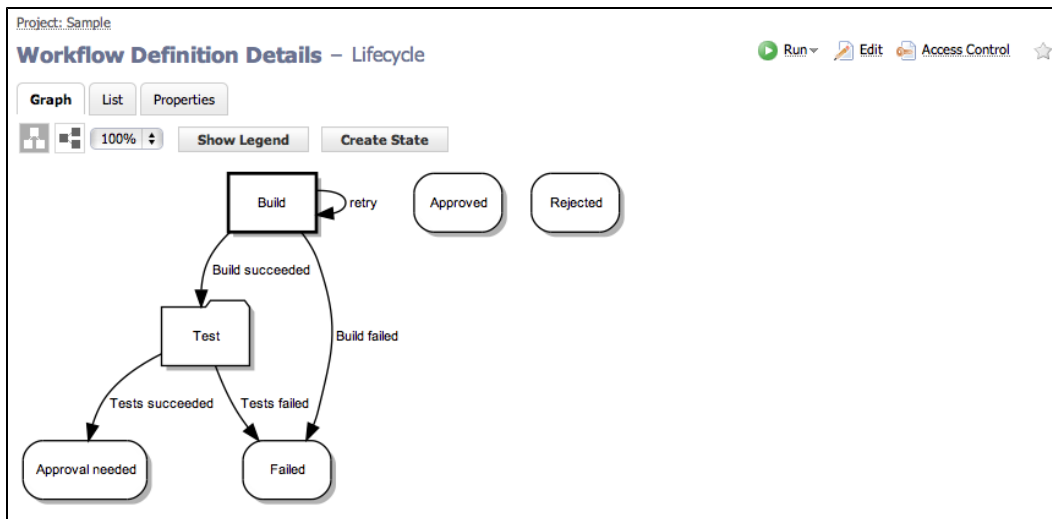
To see exactly what happened in the workflow, you can "drill-down" by clicking on the workflow link shown in the History panel.



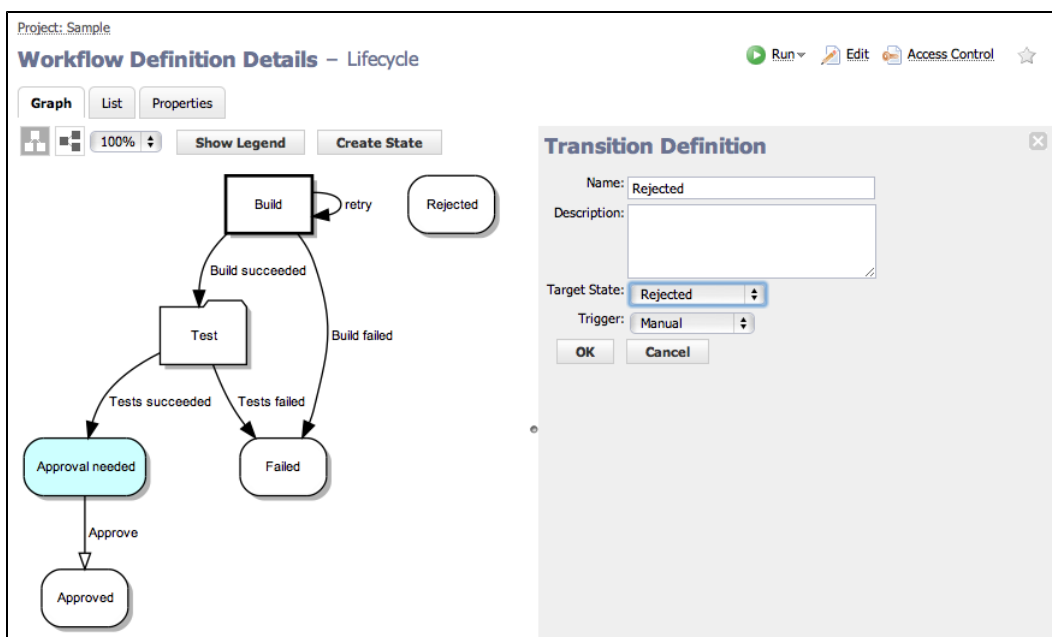
Waiting for manual intervention

Allow users to choose how to progress the workflow.

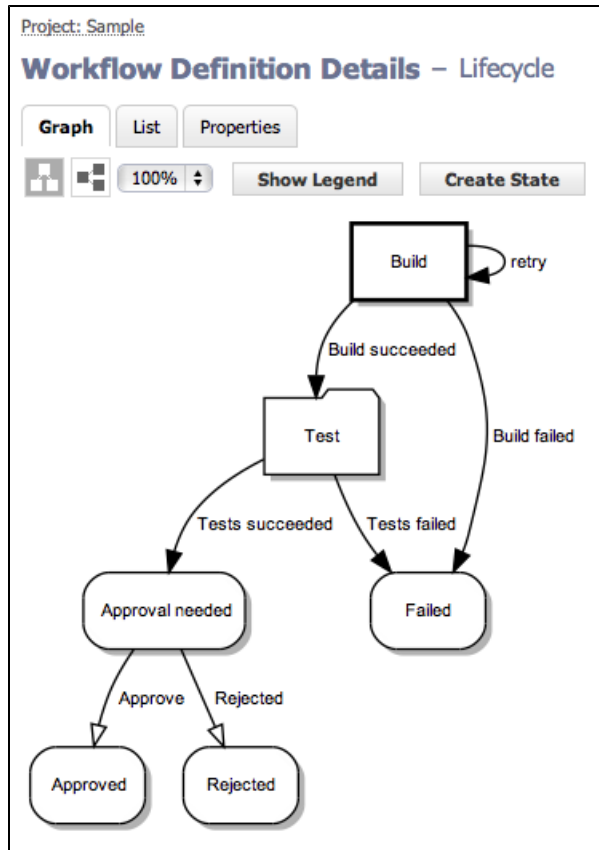
- From the Workflow Definition Details page for *Lifecycle*:
 - Create an *Approved* state and a *Rejected* state for the workflow to transition to when a user approves or rejects the build.



- For the *Approval needed* state, we create a manual transition called *Approve* that the user can select to progress the workflow to the *Approved* state.

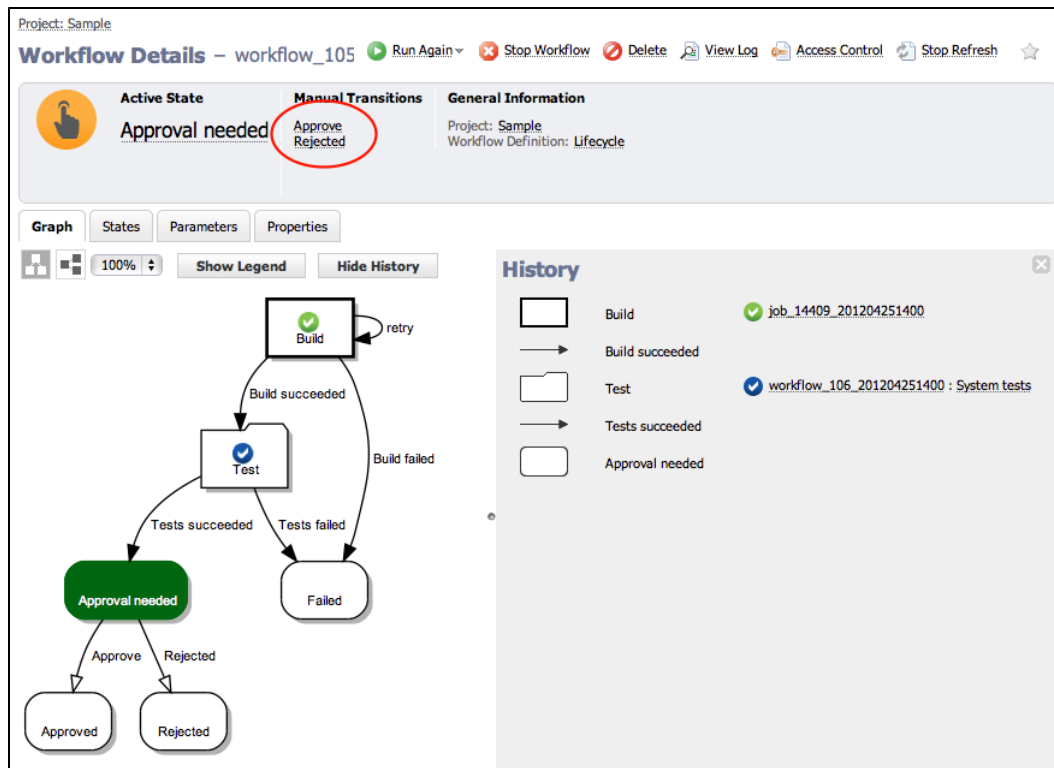


- Next, we create a manual transition called *Reject* that the user can select to progress the workflow to the *Rejected* state.
- At this point, our workflow now looks like the following example.

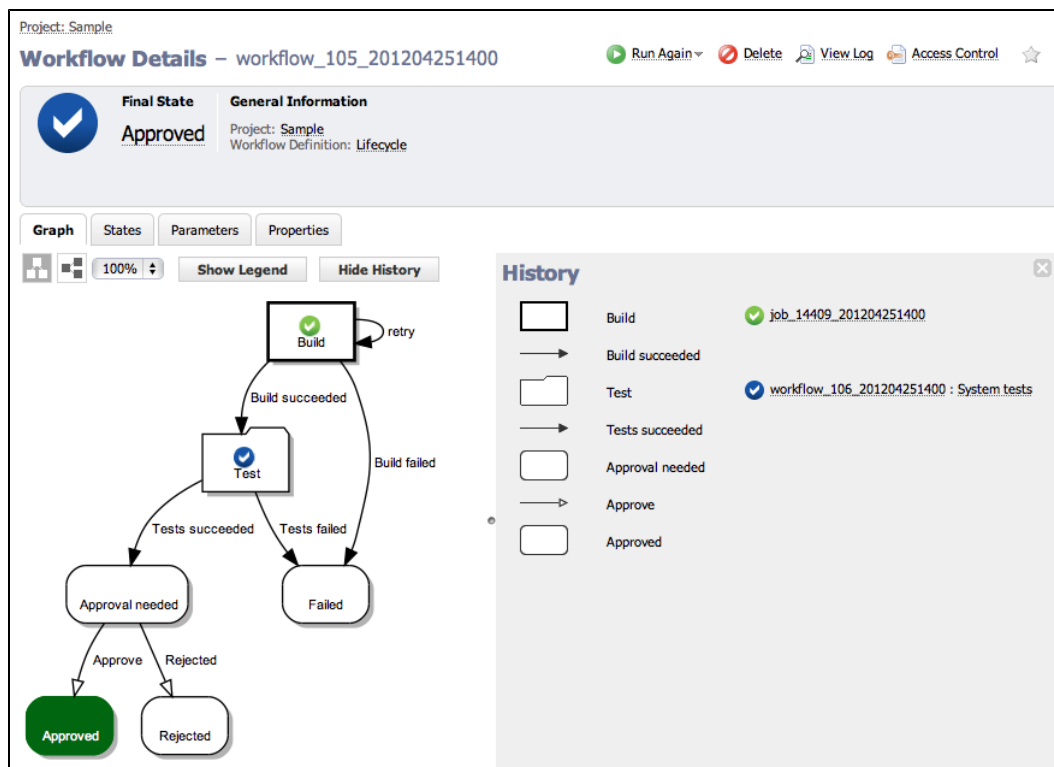


- When you run the workflow and the build and test phases succeed, the new manual transitions are made available to users on the Workflow Details page.

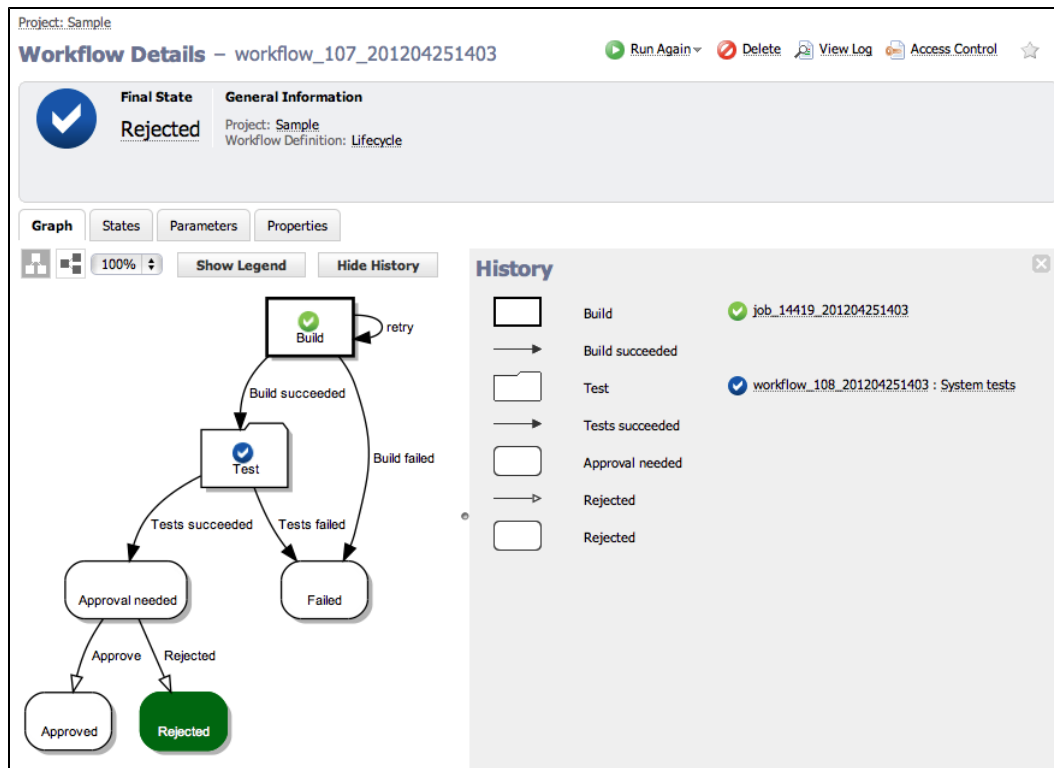
Note: If a user does not have execute permission for manual transitions, the Manual Transitions section will not be included in the summary section at the top of the Workflow Details page.



- If you choose the *Approve* transition, the workflow will end up in the *Approved* state.



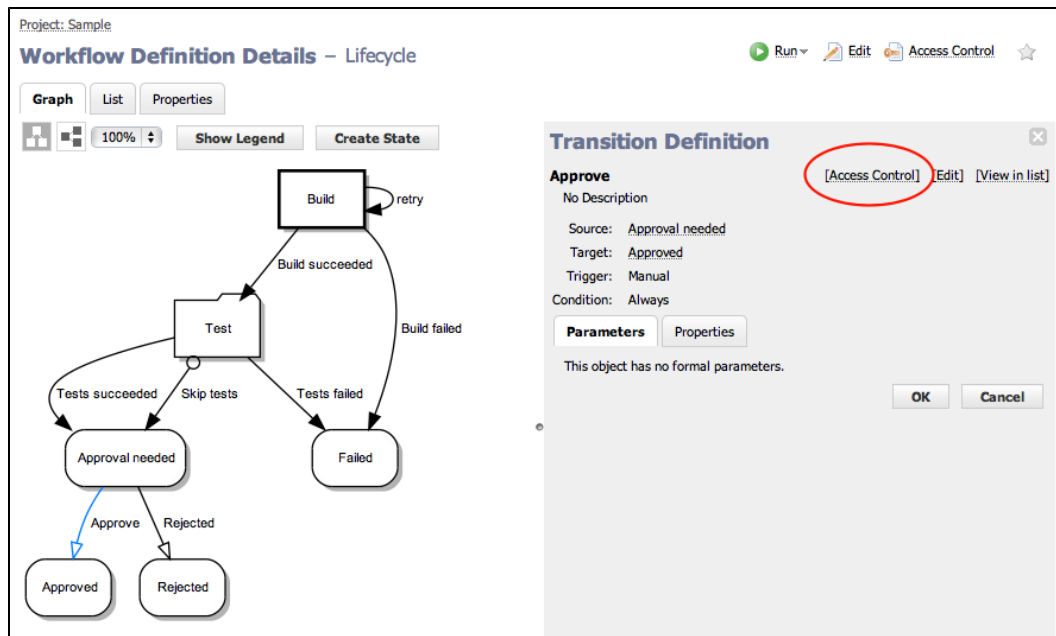
- If you choose the *Reject* transition, the workflow will end up in the *Rejected* state.



Restricting who can take a manual transition

Only allow certain users to take manual transitions.

- From the Workflow Definition Details page for *Lifecycle*:
 - When the workflow is created, access control entries from the Transition Definition are copied to the corresponding transition.
These are the access control entries for the *Approve* transition before we edit them (see below).
 - To add access control to a transition, select the transition or transition name.
 - When the Transition Definition panel opens, click the **Access Control** link,



- When the Access Control page is displayed, click the **Add Group** link to add privileges for a group that can take this transition.

Project: Sample / Workflow Definition: Lifecycle / State Definition: Approval needed / Transition Definition: Approve

Access Control – Transition Definition: Approve

Privileges for Transition Definition: Approve

Currently, there are no records to display in this list.

Add User | **Add Group** | Add Project | Break Inheritance

Privileges inherited from State Definition: Approval needed

Currently, there are no records to display in this list.

Privileges inherited from Workflow Definition: Lifecycle

Currently, there are no records to display in this list.

Privileges inherited from Project: Sample

Type	Name	Location	Read	Modify	Execute	Change Permissions
project	Sample		allow	allow	allow	allow

Privileges inherited from Projects

Privileges inherited from Server

Type	Name	Location	Read	Modify	Execute	Change Permissions
group	Everyone		allow	inherit	allow	inherit

Note: If you already have groups created, you can use them here or create two groups now: *Lifecycle Approvers* and *Lifecycle Modifiers*.

- Allow read and execute access for a group named *Lifecycle Approvers*. These users can see and take a manual transition in a workflow.
- Allow read, modify, and change permission access for a group named *Lifecycle Modifiers*. These users can edit the Transition Definition, but will not have access to take a manual transition in a workflow.

- "Break the inheritance" so only specified groups have access to this transition.
- Duplicate this access control configuration for the *Reject* transition.

Project: Sample / Workflow Definition: Lifecycle / State Definition: Approval needed / Transition Definition: Approve

Access Control – Transition Definition: Approve

Privileges for Transition Definition: Approve

Type	Name	Location	Read	Modify	Execute	Change Permissions	Actions
group	Lifecycle Approvers		allow	inherit	allow	inherit	Edit Delete
group	Lifecycle Modifiers		allow	allow	inherit	allow	Edit Delete

Transition Definition "Approve" disallows further inheritance.

- When a workflow is launched and reaches the *Approval needed* state, a user who is not a member of the *Lifecycle Approvers* group will not be allowed to execute any manual transitions.
- A user who is a member of the Lifecycle Approvers group can execute manual transitions.
- The Workflow Details page updates in real-time, which means you will see changes at the top of the page pertaining to the current state in the running workflow, and current changes in the graph also. For example, if the previous state had a manual transition, but the current state now processing does not have a manual transition, the Manual Transition section will not be displayed.

electriccommander

Logged in as 'Manager' Logout Help

Home Projects Jobs Workflows Cloud Artifacts Search Administration

Project: Sample

Workflow Details – lifecycle-3.5-workflow-8

Run Again Delete View Log Access Control

Final State Approved

General Information
Project: Sample
Workflow Definition: Lifecycle

Graph States Parameters Properties

100% Show Legend Hide History

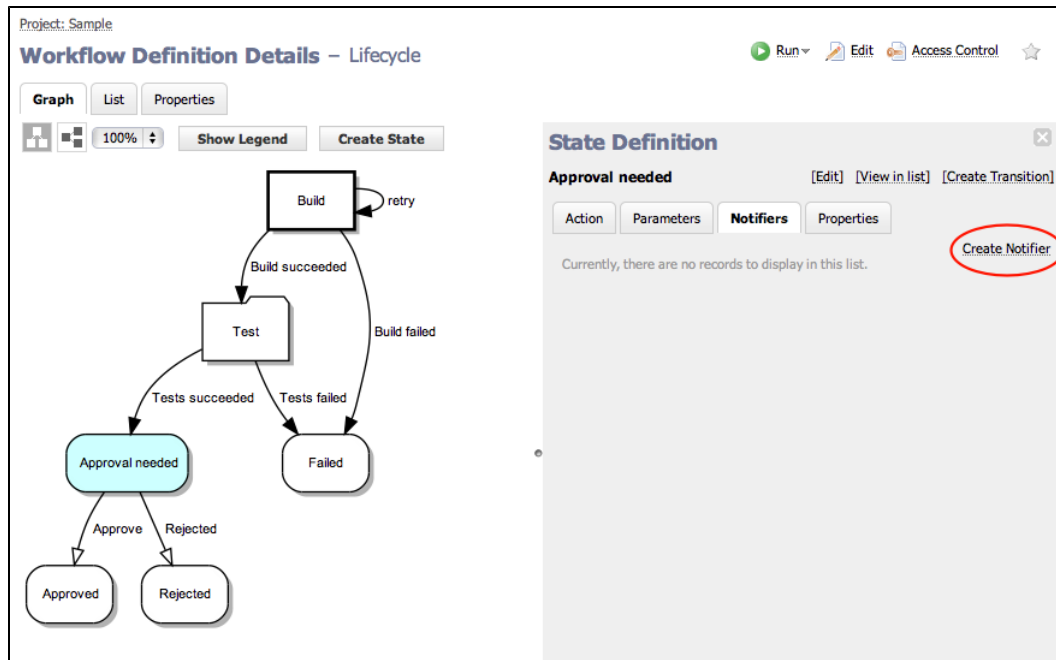
History

- Build job_19937_201204292333
- Build succeeded
- Test workflow_146_201204292333 : System tests
- Tests succeeded
- Approval needed
- Approve
- Approved

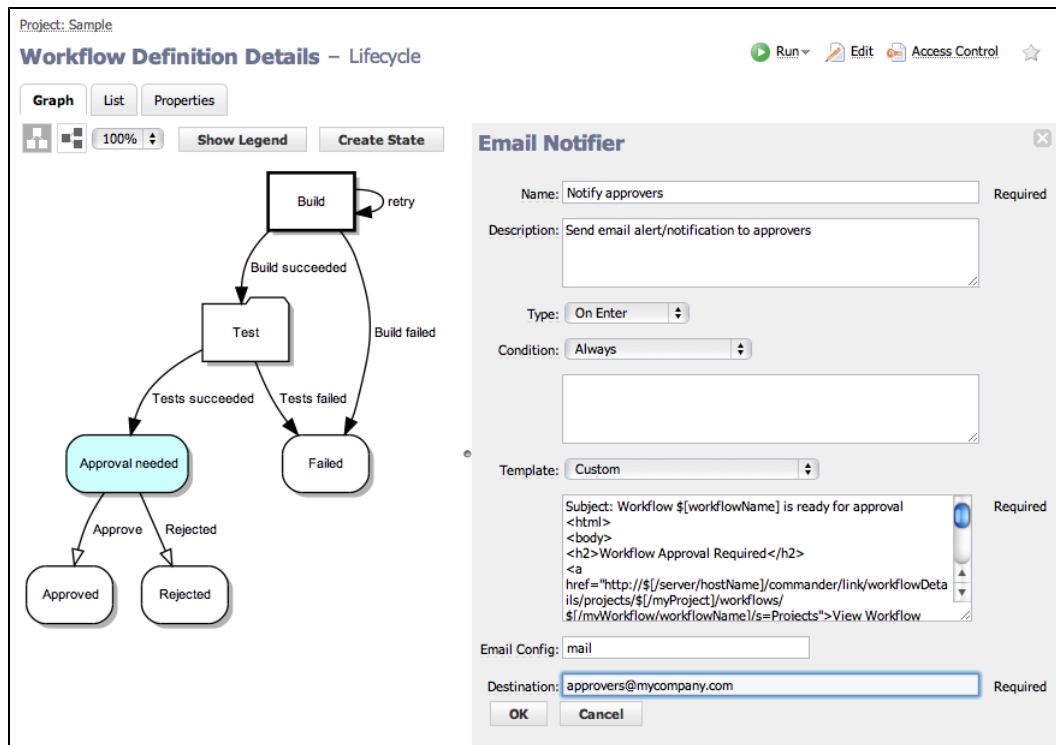
Sending email notifiers

Send email notifications so a user or group knows they need to take an action on a workflow.

- From the Notifications tab for the *Approval needed* state definition, click the Notifiers tab.
- Click the **Create Notifier** link.



- We want to create an On Enter email notifier.
- When a workflow enters the *Approval needed* state, this email notifier will be sent out.



- In this example, we want to send an email notification to the "approvers". The email will contain a link to the Workflow Details page and links to all available transitions.
- Use this formatting template for the email notifier: http://docs.electric-cloud.com/commander_doc/5_1/PDF/WorkflowTutorialScript.txt.

It includes a link to the ElectricFlow workflow, embeds a block of JavaScript that cycles through all the manual transitions from the active state, and has a link to go directly to those transitions. To ensure that the script works properly, use the script as is and make sure there are no white spaces before `<!DOCTYPE...>`, `<html>`, and `</html>`.

- This script is a sample of the type of script you can insert into the Formatting Template field illustrated in the following screen example.

- When a workflow is run and it reaches the *Approval needed* state, an email similar to the following example can be sent to specified destinations.

Only users who have execute permission on these transitions will be able to approve them by clicking the link.



Adding a global parameter to use later in the workflow

Prompt for a parameter when the workflow is launched, and use its value later in the workflow.

- Currently, there is one parameter for the starting state *Build*.
- Click on the *Build* state in the graph and select the Parameters tab.
- Click the **Create Parameter** link to create a second parameter.

Project: Sample

Workflow Definition Details – Lifecycle

Run Edit Access Control

Graph List Properties

100% Show Legend Create State

State Definition

[Edit] [View in list] [Create Transition]

Build

Action Parameters Notifiers Properties

Create Parameter

Name	Default Value	Type	Req?	Description	Actions
branch		entry	✓	The branch of the product to build	Delete

- Add a checkbox-type parameter named *skipTests* which, if selected, should skip the test suite after the build is completed.

Project: Sample / Workflow Definition: Lifecycle / State Definition: Build

New Parameter

Name:

Description:

Type: ☐ Checkbox

Value when unchecked:

Value when checked:

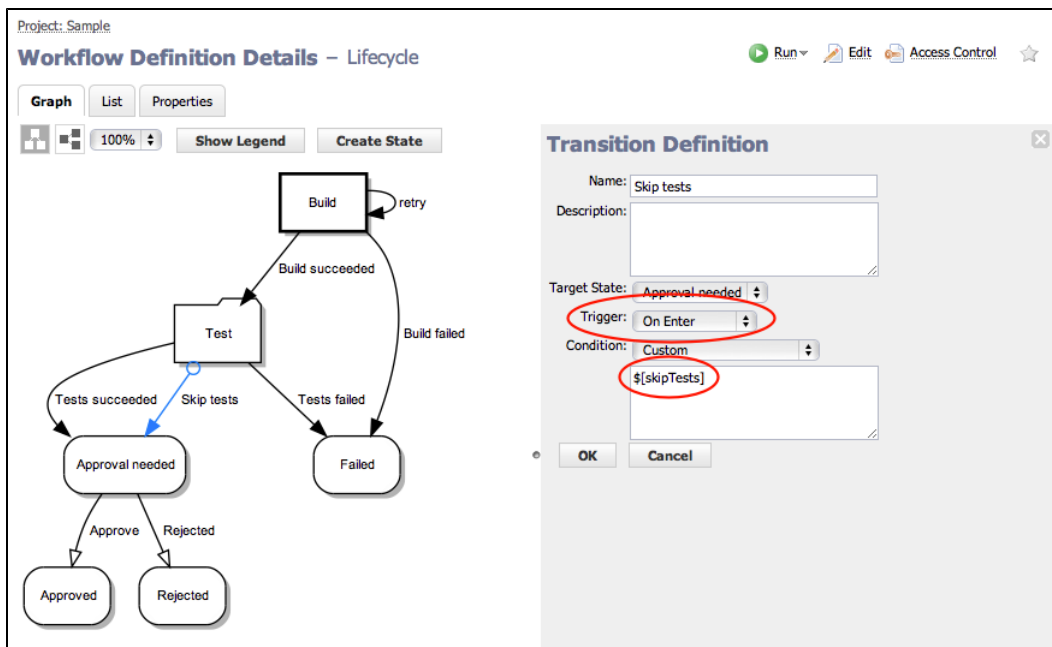
Initially checked? ☐

Default Value:

Required? ☐

Defer Expansion? ☐

- Create an On Enter transition from *Test* to *Approval needed* if the *skipTests* parameter evaluates to "true".
Because this is an On Enter transition, it will be taken before the automated tests workflow is started.



- When the workflow is launched, you will see the *skipTests* checkbox.

Project: Sample / Workflow Definition: Lifecycle

Run Workflow – Lifecycle

Starting State: Build

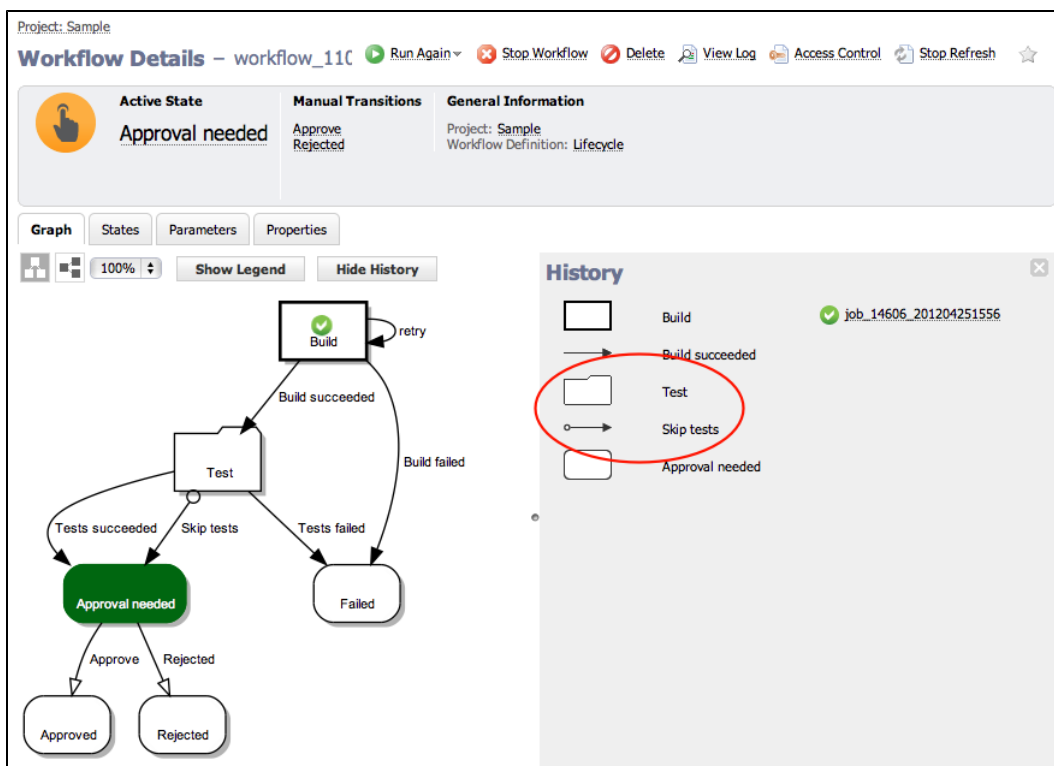
Parameters: branch: 3.5 Required

skipTests: ☒

OK Cancel

- If the parameter is checked, and after the *Build* state is completed, the workflow will immediately transition from *Test* to *Approval needed*, without starting the subworkflow.

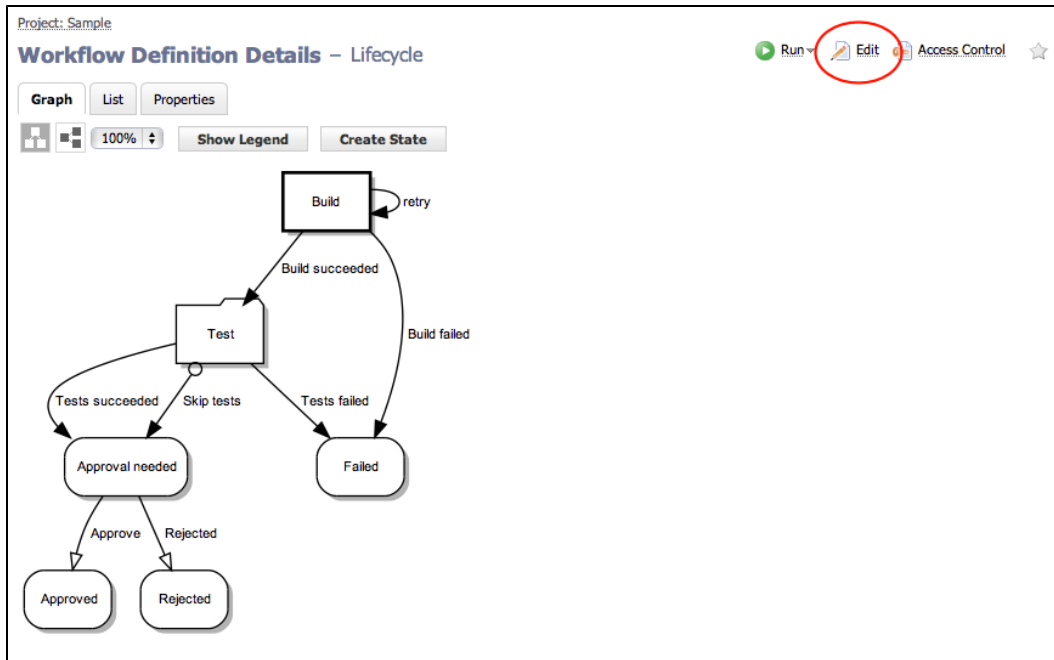
Note: Looking at the workflow History, we see the *Test* state was entered, but the subworkflow (and subjobs) was not executed.



Setting the name of your workflow

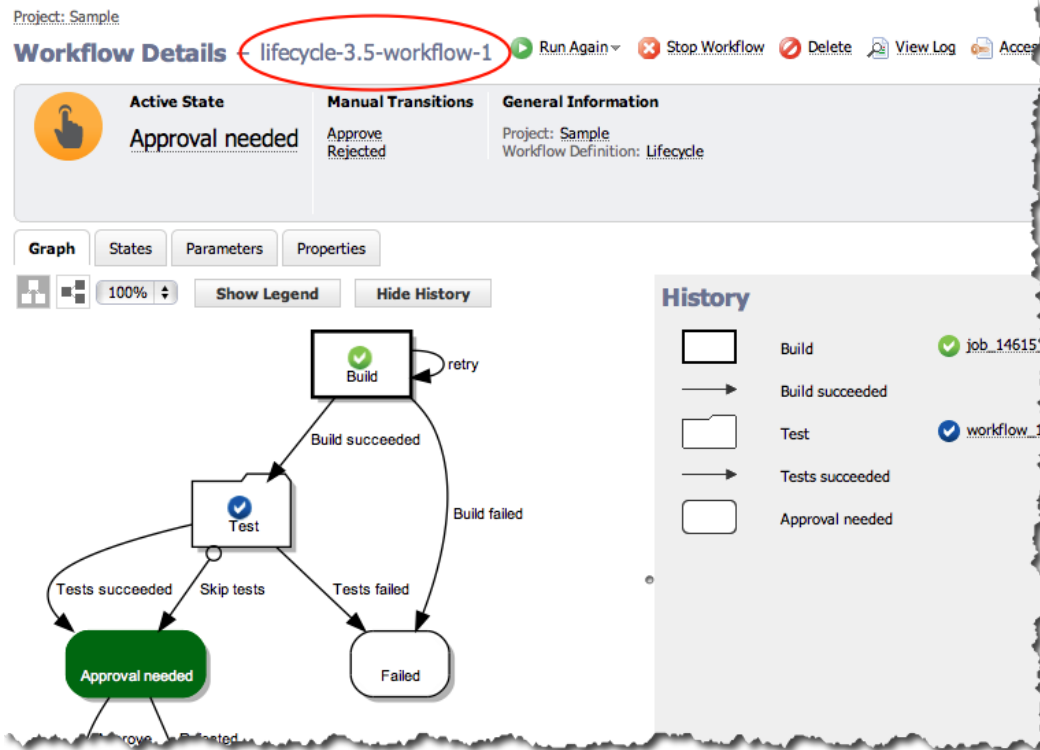
Choose appropriate names for workflows so they are easy to identify.

- If the Workflow Name Template was not set, workflows are created with a default name (workflow <jobId> <timestamp> or workflow_239_201012091208, for example). However, because you may have many different workflow definitions, you might want to create a distinguishing name for workflows launched from each definition.
- From the Workflow Definition page for *Lifecycle*, click the **Edit** link at the top of the page.



- Setting the Workflow Name Template (as shown in the sample below) achieves this goal.
- `lifecycle-${branch}-workflow-${/increment /myProject/lifecycle-counter}`
- `${branch}` refers to the value of the branch parameter as specified when the workflow is launched.
- `${/increment /myProject/lifecycle-counter}` creates a property called `lifecycle-counter` on the project and increments the property every time a new workflow is launched so the names are unique.

- Now, when the workflow is launched, its name is much more descriptive than the default workflow name.



Workflow List

Most of the time, you will want to interact with your workflow from the default Graph view. However, if you want to see your workflow in a list or table view, click the List button. See the [Workflow Definition Details](#) Help topic for detailed information about this page.

Project: Sample

Workflow Definition Details – Lifecycle

Run Edit Access Control

Graph **List** Properties

Create State Definition

Name	Type	Action	Startable	Description	Actions
Build	Subjob	Sample : BuildProduct	✓		Create Transition Copy Move Delete
retry	On Completion	Target: Build		Retry if the Build state fails	Access Control Copy Move Delete
Build succeeded	On Completion	Target: Test			Access Control Copy Move Delete
Build failed	On Completion	Target: Failed			Access Control Copy Move Delete
Test	Subworkflow	Sample : Automatic Tests			Create Transition Copy Move Delete
Tests succeeded	On Completion	Target: Approval needed		Tests from the sub-workflow have completed with success	Access Control Copy Move Delete
Tests failed	On Completion	Target: Failed			Access Control Copy Move Delete
Skip tests	On Enter	Target: Approval needed			Access Control Copy Move Delete
Failed	No action			An error occurred while the workflow was running	Create Transition Copy Move Delete
Approval needed	No action				Create Transition Copy Move Delete
Approve	Manual	Target: Approved			Access Control Copy Move Delete
Rejected	Manual	Target: Rejected			Access Control Copy Move Delete
Approved	No action				Create Transition Copy Move Delete
Rejected	No action				Create Transition Copy Move Delete

The transition order is important because transitions are evaluated in order and the first one with a condition that evaluates to 'true' is taken. In the next screen example, the *Test* state has two On Completion transitions. When the state completes, it evaluates the *Tests succeeded* transition first. If a third transition was added for the *Test* state, you could use this List view and Move the transition to the correct order for evaluation.

Project: Sample

Workflow Definition Details – Lifecycle

[Edit](#) [Access Control](#) [☆](#)

[Graph](#) **List** [Properties](#)

[Create State Definition](#)

Name	Type	Action	State	Actions
Build	Subjob	Sample : BuildProduct		View Transition Copy Move Delete
retry	On Completion	Target: Build		Access Control Copy Move Delete
Build succeeded	On Completion	Target: Test		Access Control Copy Move Delete
Build failed	On Completion	Target: Failed		Access Control Copy Move Delete
Test	Subworkflow	Sample : Automatic Tests		View Transition Copy Move Delete
Tests succeeded	On Completion	Target: Approval needed		Access Control Copy Move Delete
Tests failed	On Completion	Target: Failed		Access Control Copy Move Delete
Skip tests	On Enter	Target: Approval needed		Access Control Copy Move Delete
Failed	No action			View Transition Copy Move Delete
Approval needed	No action			View Transition Copy Move Delete
Approve	Manual	Target: Approved		Access Control Copy Move Delete
Rejected	Manual	Target: Rejected		Access Control Copy Move Delete
Approved	No action			View Transition Copy Move Delete
Rejected	No action			View Transition Copy Move Delete

Another workflow example

An advanced sample Workflow Definition is pre-defined for you in the EC-Examples project. This is a "working" example, also modeling the workflow process, with which you can interact. To navigate to this example: Select the Projects tab > EC-Examples > Workflow Definitions subtab > then select Build Workflow With Approvals.

Workspaces and Disk Space Management

[Defining workspaces](#)

[Using workspaces](#)

[Workspace directory names](#)

[Working directories](#)

[Workspace accessibility](#)

[Local workspaces](#)

[Access control](#)

[Impersonation and workspaces](#)

[ElectricFlow managed files](#)

[Disk space management](#)

[ecremotefilecopy](#)

Overview

ElectricFlow provides each job step with an area on the disk it can use for "working files" and results. This disk area is called a **job workspace**. (The job workspace defaults to a directory under the workspace directory.)

- A job step can create whatever files it needs within its workspace, and ElectricFlow automatically places files in the workspace, such as step logs. Normally, a single workspace is shared by all steps in a job, but it is possible for different steps within a job to use different workspaces.
- The location of the job step workspace is displayed on the Job Details page for the job under "Details" for the step.
- Workspaces are grouped together, with workspaces for different jobs allocated as children of the same *workspace root*. When you define procedures, you can specify which workspace roots to use, then ElectricFlow creates a child directory within that workspace root for the job's steps to use.
- The term "workspace" is used at different times to refer to either a job workspace or a workspace root. Depending on the context, it should be easy to discern which "workspace" is being discussed or referenced.

Defining Workspaces

ElectricFlow can handle any number of workspace roots, which are defined by selecting the Cloud > Workspaces tabs on the web interface. To create a workspace root, provide a workspace name and three file paths for jobs to access the workspace:

Workspace name

The workspace name must not have trailing spaces.

When two workspaces have similar names, such as "Server1" and "Server1 " (with an extra space at the end), the database records them as two entries, one without trailing spaces and one with trailing spaces.

If you run a job against one of these workspaces, it may hang and not proceed because ElectricFlow is waiting for a resource, an available workspace, or a step to complete.

UNC path

A UNC path that Windows machines can use to access the workspace via the network, such as

`//server/share/x/y/z.`

You must ensure this path is accessible on any resource where the workspace is used.

Drive path

A Windows path, using a drive letter that refers to the same directory as the UNC path. Before running a job step using the workspace, the ElectricFlow agent creates a drive mapping for this path to match the UNC path.

For example, if the UNC path is `//server/share/x/y/z` and the drive path is `N:`, the agent will map `N:` to `//server/share/x/y/z`. If the drive path is `N:/y/z` then the agent will map `N:` to `//server/share/x`.

The directories in the drive path after the drive letter (`y/z` in the preceding example) must match the last directories in the UNC path.

UNIX path

A path used on UNIX machines to access the workspace via the network, typically using an NFS mount. You must ensure appropriate mounts exist on all resources where the workspace is used.

If your environment contains Windows machines only, you can omit the UNIX path, and if your environment contains UNIX machines only, you can omit the UNC and Drive paths.

During the ElectricFlow installation, you had the option of defining a workspace. If you defined a workspace, it was named "default" and will be the default workspace for all jobs, as described below.

By default, remote workspaces are accessed with the agent user's credentials. For Windows resources, you can override that user's credential by attaching a credential to the workspace. Using a credential attached to a workspace is useful if the agent machine is not a member of the domain, but the workspace is located on a machine in the domain. An attached credential for a domain user allows the agent to manipulate the workspace as that domain user.

Using Workspaces

The simplest way to use workspaces is to define a single workspace named "default" that is readable and writable on all resources, and never specify any other workspace information. With this approach the default workspace will be used for all steps in all jobs.

However, you can also arrange for different jobs to use different workspaces, or even for different steps within a single job to use different workspaces.

You can specify a workspace name in any of the following ElectricFlow objects:

- If you specify a workspace in the definition of a procedure step, it applies to that step.
- If you specify a workspace in the definition of a procedure, it applies to all steps in the procedure.
- If you specify a workspace in the definition of a project, it applies to all steps in all procedures defined in that project.
- If you specify a workspace in the definition of a resource, it applies to all steps assigned to that resource.

If workspaces are defined in several of these places, the workspace for a particular step is chosen in priority order corresponding to the list above: a workspace specified in a step takes priority over all others, followed by a workspace in the procedure, project, and finally resource. If no workspace is specified in any of these locations, the workspace named "default" is used.

Workspace Directory Names

A job workspace directory name, within its workspace root, is derived from the name of the job.

For example, if a job named "job_63" uses a workspace whose root directory is `//ec/workspaces`, the job workspace directory will be `//ec/workspaces/job_63`. Changing the name of a job has no impact on workspace names: the initial name of the job is always used for workspace directory names.

A job is allocated a single job workspace only within a particular workspace root, which will be shared by all steps that specify that root.

For example, suppose a job has 5 steps: 3 steps specify workspace A and 2 steps specify workspace B. The 3 steps that specify A will all share a single directory under A, and the other 2 steps will share a single directory under B.

Working Directories

By default, each job step begins execution with its current working directory set to the top-level directory in the job's workspace. A job step can override this location by specifying an explicit working directory. If a job step specifies a relative path for its working directory, the path is relative to the top-level workspace directory.

Workspace Accessibility

The simplest way to set up ElectricFlow is to make every workspace root accessible on every resource. This accessibility mode provides the most flexibility and simplifies ElectricFlow system management.

Unfortunately, some environments cannot allow such universal access.

For example, some sites do not provide file sharing between UNIX and Windows machines. In large enterprise environments where ElectricFlow is shared among multiple groups, you may want to partition resources among the groups and limit access by each group to the other groups' resources. In the most extreme case, you may not have network file sharing at all. ElectricFlow can handle all of these cases.

If a workspace root is not accessible on a particular resource, you should not use that workspace in any step that runs on the resource. If you do, the resource will not be able to create the job workspace directory and the step will stall until the issue is resolved. A workspace root (on a resource) must be both readable and writable to be used for job steps.

Local Workspaces (Disconnected Workspaces)

Normally, workspaces are shared, which means if the same workspace is used on different resources, they see the same set of files. However, it is possible to define a *local* workspace (also referred to as a disconnected workspace), which means it will refer to a different local directory on each resource where it is used.

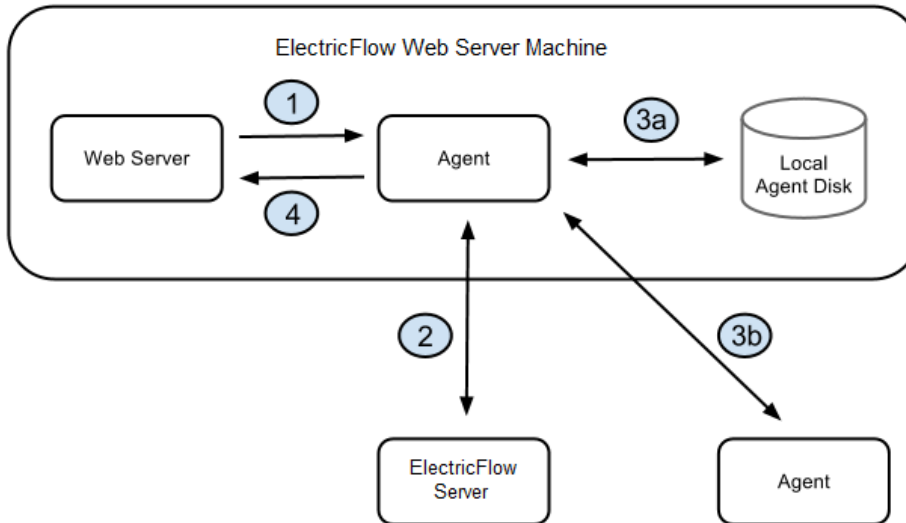
To define a local workspace on Windows, specify a local path such as `C:/Windows/Temp/ecworkspace` for the drive path.

For UNIX, specify a local UNIX path such as `/usr/tmp/ecworkspace`. Local workspaces are advantageous because they do not require remote network access and access may be faster in some cases.

However, you cannot share workspace information among job steps running on different resources.

Note: Web servers "ask" the relevant agent for access to log files in workspaces. The agent streams the file back to the web server, which then streams the log file to your browser. However, this mechanism works only for Windows and Linux 4.2 (or later) agents. Solaris, MacOS, and HP-UX agents must transfer log files to a

workspace proactively so a Linux or Windows agent can access them for the web server—this task can be made easier, using the `ecremotefilecopy` tool. See the [ecremotefilecopy](#) section in [ElectricFlow Installed Tools](#) on page 602 for details.



The diagram illustrates the architecture when the web server requests a log file and you have defined a local workspace.

1. The web server asks the local agent to retrieve the file from a workspace.
2. The agent (local to the web server) asks the ElectricFlow server how to find the workspace, and the ElectricFlow server replies with a route to an agent and validates the session ID.
3. The route to the agent that has the file could be: (a) the local agent itself or (b) a different agent that ran the step that produced the file.
 - a. The agent (local to the web server) gets the file from its local disk.
- or**
- b. The agent gets the file from some other agent.
4. The agent (local to the web server) sends the file to the web server.

Access control

You can restrict access to workspaces using the ElectricFlow access control mechanism. Before a step can use a workspace, it must have the "execute" privilege on the workspace, where "it" means the ElectricFlow user ID under which the step executes (the project principal). You can set permissions on a workspace to enable or disable access as you choose.

Impersonation and workspaces

Before a job step can use a particular workspace, three access limitations must be satisfied:

The job step must have execute access to the workspace object.

If not, ElectricFlow aborts the step with an access control violation. See the [Access Control](#) Help topic for details.

The workspace must be accessible on the resource where the step runs.

If not, the ElectricFlow agent is unable to create the step's log file and the step's execution will fail. See [Workspace Accessibility](#) (above) for more information.

The Windows or UNIX agent account and/or job step must have read/write access to workspace files.

If not, the step's execution will fail in one of several ways. This issue is the topic of this section.

Potentially, two system accounts can impact each job step execution:

The first account is the one used by the ElectricFlow agent—you selected this account when ElectricFlow was installed. The agent account must have the ability to write the workspace root directory (to create the job workspace directory), and it must have the ability to write the job workspace directory to create log files.

The second account is the one under which the step executes. If the step does not use user impersonation (in other words, no credential has been specified for it), the step runs under the same account as the agent so there are no additional issues. However, if the step is running with credentials, the credentialed account must also have read/write access to the workspace area.

Because of these requirements, you may end up with a configuration where any job step running in a workspace can read or write any file under that workspace root, including files from other jobs that share the same workspace root. In many environments this solution works, but in some situations you may want to prevent one job from accessing files from a different job.

To create job privacy, run all of the job's steps with credentials for a separate account. In addition, run a job step at the beginning of the job that creates a subdirectory within the job workspace and change the protection on that directory to permit access to the credentialed account only. Next, place all private files for the build in the protected subdirectory. These files will not be accessible to other jobs or the agent. The top-level directory in the job workspace still needs to be accessible to the agent so it can read and write log files, but everything in the lower-level directory will be private.

ElectricFlow managed files

ElectricFlow automatically places certain files in the top-level job step workspace directory:

Step logs

When a step's command executes, its standard output is redirected to a log file unique to that step. The log filename is derived from the step name and its unique identifier within ElectricFlow.

For example, a step named "step1" will have a log file with a name like "step1.2770.log", where 2770 is a unique identifier assigned by ElectricFlow. A unique identifier is needed in situations where the same step name occurs multiple times in a job such as multiple invocations of a single nested procedure.

Postprocessor logs

If a step specifies a postprocessor, standard postprocessor output is redirected to a file in the workspace. The postprocessor output file will have a name similar to the step log. For example, "step1.2770-postp.log".

Diagnostic extracts

If a postprocessor extracts diagnostic information from a step's log file, it usually places that information in an XML file in the top-level job workspace directory, and then it sets a property that contains the filename. The ElectricFlow *postp* postprocessor uses filenames like diag-2770.xml, where 2770 is the unique identifier for the step. Other postprocessors may use different filenames.

Disk space management

The current ElectricFlow version does not provide a facility for automatically deleting old workspaces. You are responsible for deleting obsolete job workspaces yourself. If you delete a job, ElectricFlow does not automatically delete its workspace. (A future ElectricFlow version may provide automated facilities for identifying and deleting obsolete workspaces—until then, you may wish to create a procedure that runs on a regular schedule and deletes old job workspaces.)

Tutorials

The ElectricFlow Help topics in this directory help you learn more about ElectricFlow and specific ElectricFlow tasks.

You can directly access these Help topics from the Help system Table of Contents.

You can also access these Help topics while using the automation platform. Click **Help** at the top-right corner on any web page in the automation platform.

Adding a Link to a Job

ElectricFlow provides an easy method for adding links to the Job Details page to view files and reports for any job where you need to access additional information quickly. Adding direct links eliminates logging into where these files may be stored.

Note: The Workflow Details page and several other ElectricFlow web pages also allow adding links .

This tutorial shows how to add links to a job and have them immediately available on the Job Details page.

To view an example procedure for this tutorial , go to the automation platform UI > **Projects** > **EC-Tutorials-<version>**, and click the procedure name.

An explanation of what you see:

- The Procedure Details page for the "Adding a link to a job" procedure contains two steps: Create File and Create Job Link

- The Action column displays a code fragment of the script used to create the `readme.txt` file and a reference for the Create Job Link step. For this example, we want to link to a "readme" file. (You can click on the **Create File** step name to go to the Edit Step page to see the full ec-perl script shown partially in the Action column.)

Click **Run** to run this sample procedure and see the resulting job status on the Job Details page, where the new link to the `readme.txt` file is displayed under the Links heading in the summary section at the top of the page.

Implementation

Use the following instructions if you would like to practice creating a link on the Job Details page now.

Note: Any changes you make within this tutorial will not be saved when you upgrade ElectricFlow. These instructions only coach you through the process you would use to create links on your Job Details pages.

1. In the automation platform UI, click **Projects > EC-Tutorials-<version> > Adding a link to a job**.
The Procedure Details page opens.
2. In the Procedure Steps section, click **Subprocedure** to create a new step based on a subprocedure.
The Choose Subprocedure Step dialog box opens.
3. To select a project:
 - a. Select **Project**. In the text box, start typing **EC-Utilities**.
As you type, possible matches appear in a drop-down list.
 - b. When you see **EC-Utilities**, select it.
4. To select a procedure:
 - a. Select **Project**. In the text box, start typing **Create Job Link**.
As you type, possible matches appear in a drop-down list.
 - b. When you see **Create Job Link**, select it.
5. Click **OK**.
The New Step page opens.
6. In the General section, Google search in the **Name** field.
7. In the Parameters section:
 - a. In the **Link Label** field, enter `Google Search`.
(This is any name you choose for the name of your link.)
 - b. In the **Link Location** field, enter `http://www.google.com`
(This is the URL to your link location, or it can be a relative path to where you want to create a link.)
 - c. Click **OK** to save your entries and return to the Procedure Details page.
8. On the Procedure Details page, click **Run**.
9. On the Job Details page, as soon as the links are processed you will see the new "Google Search" link in addition to the "readme.txt" link.

Clicking on the new Google Search link will take to you to the Google website.

Related Information

[Job Details](#) - See this Help topic for more detailed information about creating links and complete information on other aspects for using the Job Details page.

Calling a Subprocedure

The ability to create *reusable components* is a major ElectricFlow strength. Procedures in any project or plugin can be called as a subprocedure in a step for another procedure. Creating a subprocedure is one way of reusing existing functionality already implemented in ElectricFlow.

This tutorial shows how to use the step configuration process to call an existing procedure to use as a subprocedure.

To view an example procedure for this tutorial, go to the automation platform UI > **Projects > EC-Tutorials-<version>**, and click the procedure name.

An explanation of what you see:

- The Procedure Details page for the Calling a subprocedure procedure contains a step named Execute a procedure from this project.
(Clicking on this step takes you to the Edit Step page.)
- The Action column displays a plugin (or project) name, EC-Tutorials, and a procedure name, Working with properties in a stored procedure.
(Clicking on the plugin/project name takes you to the Project Details page for that project, and clicking on the procedure name takes you to the Procedure Details page for that procedure.)

Click **Run** to run this sample procedure and see the resulting job status on the Job Details page.

Implementation

Use the following instructions if you would like to practice creating another subprocedure for this "Calling a subprocedure" procedure.

Note: Any changes you make within this tutorial will not be saved when you upgrade ElectricFlow and cannot be transferred to your procedures. These instructions only provide a clearer explanation for how you can call subprocedures in your projects.

To create a step that executes a subprocedure:

1. On the Procedure Details page, click the **Create Step** link to see the Choose Step Type dialog.
 - Select the Subprocedure tab.
 - Select Plugin and place your cursor in the text field and start typing EC-Tutorials.
 - For Procedure, click inside the text box and select Reserving a Resource for job step duration procedure or whichever procedure you would like to choose.
 - Click **Create** to go to the New Step page to create the step to call an existing procedure to use as a subprocedure.
2. On the New Step page, enter any new step name of your choice.
Notice that no parameters are required for this step.

3. Click **OK** to return to the Procedure Details page to see your new step in the table, then click **Run**.
4. When the Job Details page is displayed, you can see your new step and the procedures it is calling.

Related Information

[Procedure - create new or edit existing procedure](#) - Help topic

[Step - create new or edit existing step](#) - Help topic

Checking the Outcome of Preceding Steps

ElectricFlow steps can have a *success*, *warning*, *error*, or *skipped* outcome. Sometimes it is useful to execute or skip subsequent steps based on the outcome of a previous step. Used with error handling behavior available at the step level, sophisticated flow control can be achieved easily.

This tutorial demonstrates how to see the outcome of a preceding step and use run conditions to execute steps based on that outcome.

To view an example procedure for this tutorial, go to the automation platform UI > **Projects** > **EC-Tutorials-<version>**, and click the procedure name.

An explanation of what you see:

The Procedure Details page for the procedure, "Checking outcome of preceding step", contains 3 steps:

- `step 1 - force an error` - This step forces an error by attempting to run on a resource that does not exist.
This action was achieved by hard-coding the step named `"step 1 - force an error"` to run on a resource called `"this-does-not-exist"`.
- `step 2 - execute if step 1 failed` - This step uses JavaScript to check the outcome of the first step and executes if the first step failed.
You can view the JavaScript for the run condition by clicking on the step name to take you to the Edit Step page—see the Run Condition field in the Advanced section.
This statement checks the result of `"step 1"` for an error. If there is an error, `"step 2"` will execute.
- `step 3 - execute if step 1 succeeded` - This step also uses embedded JavaScript to check the outcome of the first step and executes if the first step succeeded.
This is a slight variation of `"step 2"`. Instead of looking at `"step 1"` for an error, this step checks for success and executes only if `"step 1"` succeeds.
You can view the JavaScript for the run condition by clicking on the step name to take you to the Edit Step page—see the Run Condition field in the Advanced section.

Click **Run** to run this sample procedure and see the resulting job status on the Job Details page.

Implementation

Use the following instructions if you would like to practice creating steps to check the outcome of a preceding step.

Note: Any changes you make within this tutorial will not be saved when you upgrade ElectricFlow and you cannot transfer any steps from this tutorial to your projects. These instruction are provided only to give you a clearer definition for how to recreate this process in your own projects.

- Click the **Create Step** link.
 - In the Choose Step Type dialog, choose **Command** step to go to the New Step page.
 - Enter a Name for your step.
 - In the Command(s) text box, you can use the information in the above section. "Copy and paste" the provided JavaScript, changing the step name (between the brackets) to the new step name you supplied.
- Repeat this process to create 3 steps, clicking **OK** to save each step.
- When your 3 new steps are displayed on the Procedure Details page, click **Run** to see your steps running on the Job Details page.

Related information

[Step - create new or edit existing step](#) - Help topic

Conditional Execution

Process automation frequently contains actions to be executed only if certain conditions are met. Conditional execution can be applied directly to your procedures rather than having to maintain larger scripts to handle these actions. For example, you may need to control whether or not to build on different operating system types.

This tutorial shows you how to implement simple conditional execution in a step.

To view an example procedure for this tutorial, go to the automation platform UI > **Projects** > **EC-Tutorials-
<version>**, and click the procedure name.

An explanation of what you see:

This procedure defines one parameter and then uses a run condition to check the parameter value to decide whether or not to execute a step.

- The Procedure Details page for the Conditional execution procedure contains a step named step 1. (Clicking on this step takes you to the Edit Step page.)
- The Action column displays the command to be executed.
- The Parameters section displays a parameter with values.
The parameter named "Execute step 1" was created as a checkbox.

If checked, the parameter equates to "true".

In the run condition for "step 1", the parameter is read using the following property reference notation:

```
$_[Execute step 1].
```

(Clicking on the Parameter Name takes you to the Edit Parameter page to make modifications you may need.)

Click **Run** (at the top of the page) to run this sample procedure, which then takes you to the Run Procedure page where you could cause `step 1` to execute by selecting the Execute step 1 checkbox.

- Click **Run** again to go to the Job Details page to see the job created by running this procedure.

Implementation

Review the following instructions if you would like to see how to reproduce a conditional execution step.

Note: Any changes you make within this tutorial will not be saved when you upgrade ElectricFlow. These step-by-step instructions are provided only for more detailed information so you can easily see how to create your own "conditional execution" solutions within your procedures.

- Click the **Create Parameter** link to go to the New Parameter page.
 - Enter a new parameter name.
 - Click the Type down-arrow and select Checkbox.
 - Select Initially checked?
 - For Default Value, type "true".
 - Select Required?
 - Click **OK** to save your parameter and return to the Procedure Details page.
- Click the step name to go to the Edit Step page.
 - Change the Run Condition text to check the new parameter name you supplied above.
 - Click **OK** to return to the Procedure Details page.
 - Click **Run** to go to the Run Procedure page to see 2 parameters where you can now decide if both are required or not.
 - Click **Run** to go to the Job Details page.
 - If only 1 parameter was "checked" as required, you will see the "skipped" notation in the Status column.

Related information

More complex run conditions can be implemented using inline JavaScript.

See the tutorial, "Checking outcome of preceding step" for a more complex run condition example.

Custom parameter layouts

By default, parameters for procedures are displayed in alphabetical order based on the parameter name. Using the `ec_parameterForm` property allows parameters to be displayed in a specified order, different from the default, alphabetized order by name.

This tutorial shows how to create a custom parameter layout for procedures.

To view an example procedure for this tutorial, go to the automation platform UI > **Projects** > **EC-Tutorials-
<version>**, and click the procedure name.

An explanation of what you see:

- This procedure implements two parameters.
- The Action column contains the command to execute. In this case, the action shows the `ec_parameterForm` property which was attached to the procedure.
- Notice the checkmarks in the Req? column, which indicates both parameters are "required" each time this procedure runs.

- In the Custom Procedure Properties table, notice the "formElement" values supplied to determine the new parameter order.
 - If the procedure is run **without** the `ec_parameterForm` property, the parameter input form displays parameters in default, alphabetical order:

Parameters

first parameter:

second parameter:

Click **Run**.

On the Run Procedure page, notice you are presented with a Parameter section that shows the parameters in reverse order because values supplied for the `ec_parameterForm` property were called to create this new order.

Parameters

this is the label for "second parameter"

this is the label for first parameter

The following XML creates the new order for these parameters:

```
<editor>
  <formElement>
    <property>second parameter</property>
    <label>this is the label for "second parameter"</label>
    <type>entry</type>
    <required>1</required>
  </formElement>
  <formElement>
    <property>first parameter</property>
    <label>this is the label for first parameter</label>
    <type>entry</type>
    <required>1</required>
  </formElement>
</editor>
```

Click **Run** to run this sample procedure and see the resulting job status on the Job Details page.

Implementation

Note: This tutorial is intended for viewing, but not modifications within its procedure, and any changes you might make will not be saved when you upgrade ElectricFlow.

However, in your own projects, you can use the sample XML above to change the parameter order in any of your procedures.

If you copy and paste the code samples we provided, using the `ec_parameterForm` property in one or more of your procedures, make sure you replace our example values with your parameter names, labels, and so on. If you have more than two parameters to reorder, create any number of additional `<formElement>` sections you need.

Related information

[Customizing the ElectricFlow UI /Customizing Parameters](#) - Help topic

[Step - create new or edit existing step](#) - Help topic

[Parameter - create new or edit existing parameter](#) - Help topic

Email Notifications

Use email notifiers to send information to people who need or want ElectricFlow notifications, whether or not they actually use ElectricFlow. For example, you might want to set up an email notifier to send log file error excerpts matched by *postp* to a team or individual responsible for investigating the error.

This tutorial illustrates how email notifiers can be attached to steps to automatically send an email when jobs, steps, or workflows execute.

Prerequisite: If you do not have an Email Configuration, you must create one before you can set up an email notification. An Email Configuration tells the ElectricFlow server which host to use for mail delivery. Remember the email configuration name you choose because you will need it later.

To view an example procedure for this tutorial, go to the automation platform UI > **Projects** > **EC-Tutorials-<version>**, and click the procedure name.

An explanation of what you see:

- A step named "do something" that puts a short message in the log file
- A parameter that defines the email address for where you want to send the notification. This parameter is used as the destination value for the notifier named "start notifier". This notifier sends an email when the procedure runs.
- An email notifier named "start notifier" with its Type specified as "onStart".
 - Click on the notifier name to view the notifier details.
 - Make sure the value in the Email Configuration field matches the email configuration name from the "Prerequisite" section above.
 - Click **OK** to save any changes and return to the Procedure Details page.

Click **Run**.

On the Run Procedure page, in the Parameters section, enter an email address, an address list, or email group name.

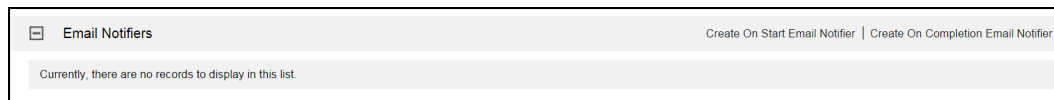
Click **Run** again to run this sample procedure and see the resulting job status on the Job Details page.

Implementation

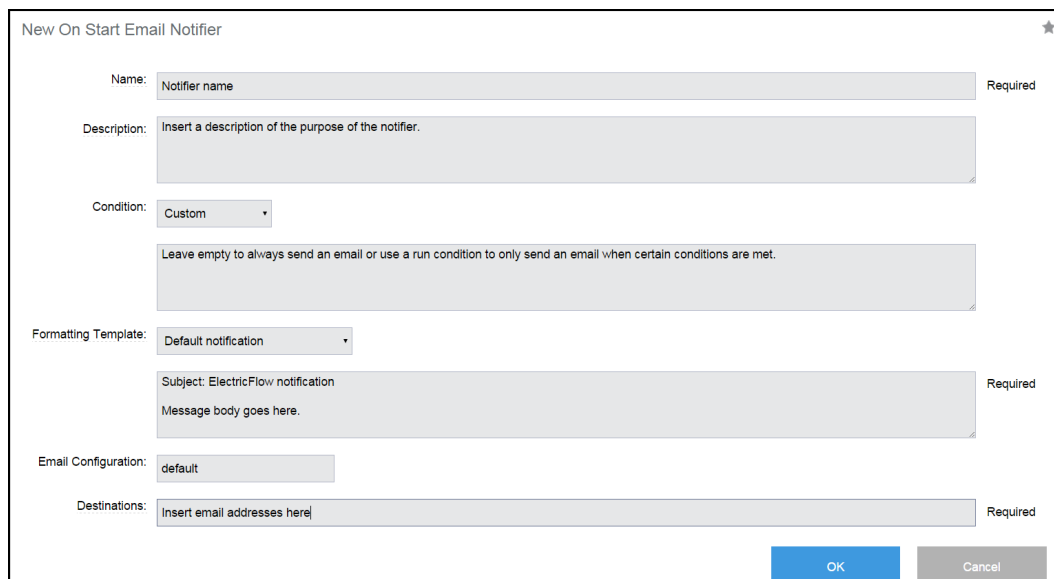
Note: Use the following information to create an email notifier in your own project, assuming you have already setup an email configuration. Any changes you might make within this tutorial will not be saved when you upgrade ElectricFlow.

Using the ElectricFlow automation platform UI to add an email notifier:

1. On the Procedure Details page, go to the Email Notifiers section, and click **Create On Start Email Notifier**.



2. Enter information in the fields on the New On Start Email Notifier page. See the example below.



Note: If the Email Configuration field is left blank, the system uses the email configuration with the name "default".

Click **OK** to save the notifier.

When the procedure is run, an email notification is sent to the email addresses specified in the Destinations field. In this tutorial example, the Destinations field refers to the parameter value passed in to the procedure when it is run.

The value is referenced using the following notation:

```
$/myJob/Send notification email to]
```

Related information

[Email Configuration - create new or edit existing email configuration](#) - Help topic

[Email Notifier - create new or edit existing email notifier](#) - Help topic

Executing Tasks on All Resources in a Pool

Creating resource pools (grouping resources) means you can reference many resources using one name, the pool name. When configuring a step, you can "broadcast" the step to all resources in a pool. For example, you might want to start an application deployment at the same time on all resources in a pool.

This tutorial shows how to group resources into a resource "pool".

To view an example procedure for this tutorial, go to the automation platform UI > **Projects** > **EC-Tutorials-<version>**, and click the procedure name.

An explanation of what you see:

This procedure implements three steps:

1. The first step, `Setup environment`, creates a pool for resources.

Clicking on the step name takes you to the Edit Step page to see the full command text supplied to create this step—a partial section is displayed in the Action column.

Note: Functionality to create this step is beyond the scope of this tutorial. For more information, see the link to "Using ectool and the ElectricFlow API" below.

The following resources were created and added to a pool named EC-Tutorial-pool:

EC-Tutorials-resource1

EC-Tutorials-resource2

EC-Tutorials-resource3

2. The second step, `Execute command on all resources in pool`, executes an echo statement on the resource EC-Tutorial-pool and has the "broadcast" option selected, which results in the step executing on all resources in the pool simultaneously.
3. The third step, `Cleanup`, deletes the resources created in the `Setup environment` step.

Click **Run** to run this sample procedure and see the resulting job status on the Job Details page.

Implementation

Note: This tutorial is intended for viewing only and/or clicking on the available links to see more information. Any modifications you might make within this tutorial cannot be transferred to your projects and will not be saved when ElectricFlow is upgraded.

Before you can create the steps illustrated in this tutorial procedure for your own purpose, you must have already defined your resources and created resource pools.

See the following list of related links for more help with creating and using resource pools.

Related information

[Resources](#) - Help topic

[Resource Pool - create new or edit existing pool](#) - Help topic

[Resource Pool Details](#) - Help topic

[Step - create new or edit existing step](#) - Help topic

Factory Procedures

Factory procedures allow dynamic behavior in process automation. To implement a factory procedure, you must first create a procedure that other procedures can use as a template. A procedure uses the template to construct new procedures at run time. Instead of creating steps or complex scripts to account for all possible scenarios, procedures can be created "on the fly" based on user input, resulting in a very dynamic system.

This tutorial shows how to create factory (dynamic) procedures, which are procedures that modify themselves at runtime based on parameters passed into them.

To view an example procedure for this tutorial, go to the automation platform UI > **Projects** > **EC-Tutorials-<version>**, and click the procedure name.

An explanation for what you see and what is actually happening:

This tutorial contains 2 procedures and creates a third procedure "on the fly":

- "Factory procedures" - this procedure uses the procedure template (Utility Procedure - Factory Template).
This procedure has a parameter called "number of steps". This parameter takes an integer and is used to specify how many new steps should be generated.
Note: For this tutorial example, only one procedure is constructed to use the template procedure, but you may have multiple procedures that use the factory template procedure.
- "Utility Procedure - Factory Template" - you do not see this procedure on this Procedure Details page, but it is included in the EC-Tutorials project. This is the factory procedure template used when generating new steps in the main procedure. This procedure implements a single step that writes some text to the log file.
This procedure is designed to do a repetitive task, repeating the task a variable number of times. For example, you might need to run a test framework "n" number of times.
Note: You can view this procedure in the EC-Tutorials project.

The main procedure, "Factory procedures", implements these steps:

- The first step: "create dynamic procedure" is a Perl script that uses the Perl API to create a new procedure.
The following script creates a new procedure containing the number of steps specified by the parameter:

```
my $procedureName = "Dynamically Created Procedure ${jobId}";  
Sec->createProcedure("${myProject/projectName}", $procedureName);  
for(my $i=0;$i<${number of steps};$i++)  
{
```



```

$ec->createStep("$[/myProject/projectName]", $procedureName, "Generated step
$i", {subprocedure => "Utility Procedure - Factory Template",

  actualParameter => {actualParameterName => 'text', value => "Executing
generated step $i"}});

}

$ec->setProperty("/myJob/dynamicProcedure", $procedureName);

```

1. First, the procedure generates a unique name for the new procedure to be created. The name is created by appending the value of the `$[jobId]` property string.
 2. A new procedure is created in the current project.
 3. Based on the number of steps to be created that were passed in as a parameter, a number of subprocedure steps are created in the procedure. The subprocedure steps call the utility procedure associated with this tutorial.
 4. The name of the newly created procedure is stored in `/myJob/dynamicProcedure`
- The second step: "Run dynamically created procedure" calls the procedure that was created in Step 1. Step 2 appears to call nothing, but in fact it is calling a procedure referred by `$[/myJob/dynamicProcedure]`.

Click **Run** to run this sample procedure and go to the Job Details page.

Running the procedure and specifying the number of steps to be generated as 3, results in the following job being executed:

Job Details / Factory procedures-4

Completed with Success

Start Time: 2015-07-12 16:45:34 PDT

Elapsed Time: 00:00:02.960

Project: EC-Tutorials-1.0.0.69904

Procedure: Factory procedures

Launched by: admin

Priority: normal

View: All

StepsDiagnosticsParametersPropertiesNotifiersPublished Artifact VersionsRetrieved Artifact Versions

Expand All | Collapse All

Step Name	Log	Status	Elapsed Time	Resource	Actions
create dynamic procedure		Completed with Success	00:00:00.893	local	<div><div></div><div></div></div>
<div>Run dynamically created procedure</div>		Completed with Success	00:00:00.720		<div><div></div><div></div></div>
<div>Generated step 0</div>		Completed with Success	00:00:00.067		<div><div></div><div></div></div>
display text	<div></div>	Completed with Success	00:00:00.075	local	<div><div></div><div></div></div>
<div>Generated step 1</div>		Completed with Success	00:00:00.060		<div><div></div><div></div></div>
display text	<div></div>	Completed with Success	00:00:00.062	local	<div><div></div><div></div></div>
<div>Generated step 2</div>		Completed with Success	00:00:00.056		<div><div></div><div></div></div>
display text	<div></div>	Completed with Success	00:00:00.068	local	<div><div></div><div></div></div>
Delete dynamically created procedure		Completed with Success	00:00:00.361	local	<div><div></div><div></div></div>

Notice the steps that execute in parallel in the dynamically created procedure:
Generated step 0, Generated step 1, and Generated step 2

Implementation

To create a "factory procedure" process in your project:

1. Create a utility procedure - this procedure performs a task.
 - Create the steps you need for your task.
2. Create a second procedure - the "factory procedure," providing a name of your choice for this procedure.
 - Create a minimum of 3 steps in this procedure.
 - step 1 - create a new procedure that contains steps, each calling the utility template procedure.
 - step 2 - this step executes the procedure created in step 1.
 - step 3 - this step deletes the procedure created in step 1.

Each time you run the "factory procedure," it creates a temporary procedure to run the utility procedure.

Note: Any changes you make within this tutorial will not be saved when you upgrade ElectricFlow.

Postp extension

postp is a powerful ElectricFlow feature (postprocessor) you can use to monitor step output in real-time and take action. Frequently, the *action* is to extract information from step output, store it in the ElectricFlow database, and set properties for reporting. In addition, *postp* can be used to execute functionality depending on whether or not a pattern is found. *Postp*, an essential part of the ElectricFlow toolbox, provides fast feedback to users about job step status.

Note: *Postp* matchers are written in Perl. *Postp* extensions are either loaded from a file located on the ElectricFlow server or from a stored property.

This tutorial demonstrates how to construct a simple custom *postp* matcher to extend *postp* capabilities, and loads the *postp* extension from a property.

To view an example procedure for this tutorial, go to the automation platform UI > **Projects** > **EC-Tutorials-<version>**, and click the procedure name.

An explanation of what you see:

- This procedure implements a single step, "get directory listing", that retrieves a directory listing using a simple Perl script. Perl is used for the example to ensure the same output no matter which operating system is used for this tutorial. Understanding the Perl code used to generate the tutorial listing is not required for this tutorial.
- A property was created. This procedure has a *postp* extension attached in a property named *matcher* and contains the following Perl code:

```
use ElectricCommander;
push (@::gMatchers,
{
  id => "fileCountMatcher",
  pattern => q{(\d+ File\s\)}},
  action => q{
    setProperty("summary", "Matcher $matcher->{id} found the following
    output\n\n$1");
  }
});
```

The `postp` matcher has an identifier ("`id`"), a regular expression ("`pattern`") that matches text and an "`action`" to be performed when text matching the regular expression is found. In this case, the regular expression "`id`" is `fileCountMatcher`, and an example of the output being matched is "`2 File(s)`". When the pattern is matched, the action to perform is to set the summary property to the text matched by the regular expression.

The custom `postp` matcher is loaded using the following command in the Postprocessor field when creating the step:

```
postp -loadProperty /myProcedure/matcher
```

Click on the step name to go to the Edit Step page to see how this step was created—note the text in the Command box and the Postprocessor field.

Click **Run** to run this sample procedure and see the resulting job on the Job Details page.

See the Status column. The number of files in the file system root where the workspace is located is displayed.

Implementation

To try using `postp` functionality in your own project:

- Edit an existing procedure or create a new one with a "`matcher`" as a property.
For your practice example, you can "copy and paste" the matcher example provided in this tutorial (above).
- Create a new Command step for this procedure - filling in the Postprocessor field on the New Steps page.
Again, for practice, you can use the `postp` string provide above in this tutorial.

When you run your new procedure, you will see results similar to those you saw after running the tutorial example procedure.

Note: Any changes you make within this tutorial will not be saved when you upgrade ElectricFlow.

Related information

[Postprocessors](#) - Help topic

[Step - create new or edit existing step](#) - Help topic

Publishing and Retrieving an Artifact

To ensure traceability in processes, output needs to be versioned and tracked as it is handed from one process stage to another. With ElectricFlow Artifact Management, implementing your processes is auditable and secure.

This tutorial shows how to produce and consume artifacts as part of an ElectricFlow process.

To view an example procedure for this tutorial, go to the automation platform UI > **Projects** > **EC-Tutorials-<version>**, and click the procedure name.

An explanation of what you see - the following concepts are illustrated in this procedure:

- The first five steps are simple ec-perl scripts that create files and directories as use as an artifact.

The first step, "Create myartifact.txt file", creates a text file in a workspace that can be added as an artifact. Click this step to go to the Edit Step page to see the full "dummy" script used to set up the artifact in the Command(s) field. This script creates a file named `myartifact.txt`.

Note: There is no code specific to ElectricFlow in this step, so understanding the Perl code used to generate the text file is not necessary.

The next four steps create a directory, a subdirectory and the second text file in it.

- The "Publish artifact" step calls the EC-Artifact:Publish procedure. This step is configured to place the file `myartifact.txt`, created in the Setup step, into an artifact.

To do this, the EC-Artifact: Publish procedure requires the following information:

The artifact where you want to publish. This is in the format `Artifact Group:Artifact Key`. In this case, the name of the artifact (where you want to publish) is `EC-Tutorials:MyArtifact`.

The artifact version being published. The version is in the form of `major.minor.patch-qualifier-buildnumber` or `major.minor.patch.buildnumber-qualifier`. For this tutorial, the version is set to `1.0.[$jobId]-tutorial`. Using the `[$jobId]` property expansion results in a new artifact being created every time the procedure is run.

The repository server from which you need to retrieve the artifact. This tutorial uses a repository named `default` that was created when ElectricFlow was installed.

Note: To see how this information is specified in this step, click the "Publish artifact" step name to go to the Edit Step page.

- The "Retrieve artifact" step calls the EC-Artifact: Retrieve procedure. This step takes two required parameters and several optional parameters (see the Edit Step page/Parameter section for this step).

The required parameters are:

Retrieve to directory – the location where the artifact files are downloaded when they are retrieved. This is where the retrieved artifact versions are stored.

Overwrite – the default is update where the artifact files are updated when they are retrieved. The other values are true (the files for the retrieved artifact versions are overwritten) or false (the files for the the retrieved artifact versions are not overwritten).

The **Retrieved Artifact Location Property** parameter is the name or property sheet path that the step uses to create a property sheet. The default value is

`/myJob/retrievedArtifactVersions/${assignedResourceName}`.

This property sheet has information about the retrieved artifact versions, including their location in the file system. It displays the location from where the artifact is to be retrieved and retrieves the property value in the `/myJob/retrievedArtifactVersions/${assignedResourceName}`.

For the **Filter(s)** parameter, enter search filters, one per line, applicable to querying the ElectricFlow database for the artifact version to retrieve.

For more information about these parameters, go to [Retrieve Artifact Version Step](#) on page 704.

- The "Output location that artifact was retrieved to" step prints the message "Artifact extracted to: <directory where the retrieved artifact will be located>"
- Click **Run** to run this sample procedure and see the resulting job on the Job Details page.

Implementation

To publish and retrieve artifacts, apply these concepts to your artifact projects, creating the steps as required to your procedure.

Note: Any changes you make within this tutorial will not be saved when you upgrade ElectricFlow.

Related information

[Artifact Management](#) - Help topic

[Artifact Details](#) - Help topic

[Job Details](#) - Help topic

[Publish Artifact Version step](#) - Help topic

[Retrieve Artifact Version step](#) - Help topic

Reserving a Resource for Job Step Duration

Sometimes you may want a resource to perform work for only one job at a time. Because a ElectricFlow agent is capable of executing steps for many jobs on a resource, this can produce undesirable performance results. For example, you might want a dedicated resource to stress test an application after you have ElectricFlow initiate a load testing task.

This tutorial shows how to reserve a resource for one job step only while that job step is running.

To view an example procedure for this tutorial, go to the automation platform UI > **Projects** > **EC-Tutorials-<version>**, and click the procedure name.

An explanation of what you see:

- This procedure locks the resource for the step and executes the tasks.
- To demonstrate this functionality, two steps are implemented.
 - Notice that both steps are set to run in parallel on the local resource.
 - Both steps request a lock of the resource for the duration of the step execution.

The functionality implemented in each step:

- A simple Perl sleep statement to force a lock over a period of time when the procedure is run
- Select "Step" in the Retain Exclusive field in the Advanced section on the New/Edit Step page

Click a name in the Step Name column (to go to the Edit Step page for that step). You will see the Perl sleep statement added to the Command text box, and "Step" selected on the Retain Exclusive field.

Instead of running the steps in parallel as you may have previously requested, the steps will now execute serially. The second step must wait for the resource to become available because it was reserved by the first step.

Click **Run** to run this sample procedure and see the resulting job on the Job Details page.

Implementation

Create or edit Command Steps for your procedures if you need this functionality.

- Go to the Procedure Details page for the procedure where you want a step to reserve a resource.
- Add a Command Step to your procedure (use the step creation link at the top of the table), or edit an existing step.
- In the Command text box, enter a script if you want to specify any time constraints.
- On the New/edit Steps page, in the Advanced section, choose "Step" in the Retain Exclusive field.

The next time you run this procedure, your job step will have exclusive use of the resource you specified.

Note: Any changes you make within this tutorial will not be saved when you upgrade ElectricFlow.

Related information

[Step - create new or edit existing step](#) - Help topic

Running Steps in Parallel

The main advantage to running steps in parallel is increased performance. You can set a step to run serially as a synchronization point and then execute more steps in parallel. For example, running steps to execute tests on different platforms concurrently would mean numerous tests could complete in the same time period.

This tutorial demonstrates how to run steps in parallel.

To view an example procedure for this tutorial , go to the automation platform UI > **Projects > EC-Tutorials-<version>**, and click the procedure name.

An explanation of what you see:

This procedure contains 5 steps:

- Step 1 and 2 run in parallel.
- The third step (sync) is a sync point where the procedure waits until steps 1 and 2 finish executing.
- Steps 4 and 5 execute in parallel after the sync point.
- Steps 1, 2, 4, and 5 execute simple Perl scripts that print a message and "sleep" for a period of time.

Note: Using "sleep" is not required, but used here to make sure each parallel step runs long enough to be visible while running this tutorial.

Click **Run** to run this sample procedure and see the resulting job status on the Job Details page.

Implementation

Note: Use the following information to create steps to run in parallel in your procedures. Any changes you make within this tutorial will not be saved when you upgrade ElectricFlow.

Steps will run in parallel if you select the Run in Parallel option in the Advanced section on the Step - create new or edit existing step page.

Advanced

Precondition

Run Condition:

Error Handling:

Procedure continues, but overall status will be error ▾

Time Limit:

minutes ▾

Run in Parallel:

☒

Always Run Step:

☐

Broadcast:

☐

Retain Exclusive:

None ▾

Release Exclusive:

None ▾

Shell

ec-perl

Workspace

Working Directory:

Log File:

Related Information

[Step - create new or edit existing step](#) - Help topic

Step Timeouts and Steps that Always Run

Complex processes can be brittle so ElectricFlow provides a way to account for different errors in your procedures. You can specify timeouts, error handling methods, and steps that always run no matter what errors have occurred in preceding steps in the procedure. These facilities can make sure your build does not "hang" in a situation where you would prefer the build to continue.

This tutorial shows how to construct a procedure so processes are terminated after a set time period and how to implement a step that always runs in the event of an error (to perform cleanup).

To view an example procedure for this tutorial, go to the automation platform UI > **Projects** > **EC-Tutorials-<version>**, and click the procedure name.

An explanation of what you see:

This procedure implements 2 steps:

(Clicking on the step name takes you to the Edit Step page to see how the step was created.)

- The "force timeout" step has a 10-second timeout and is set to abort the procedure and terminate running steps in the event of a timeout or error.

This step executes a sleep command using ec-perl, which makes the step execute for 15 seconds.

Step error handling is set to "Abort procedure and terminate running steps". This step will always fail because the 15-second sleep time is longer than the 10-second timeout.

- The "always run" step set to run no matter what the error handling behavior is for any preceding steps.

This step uses an echo statement to output some text. The step property, *Always Run Step*, is set to "true" so this step runs regardless of the result for the preceding step.

Click **Run** to run this sample procedure and see the resulting job on the Job Details page.

On the Job Details page: Notice that the "always run" step completed with Success while the "force timeout" step completed with a TIMEOUT error.

Implementation

Note: This tutorial is intended for viewing only. Any changes you might make within this tutorial cannot be transferred to your projects will not be saved when you upgrade ElectricFlow.

As you create steps for your projects, you can duplicate these tutorial concepts. On the New Steps page, review the Advanced section. Notice the Error Handling and Always Run Steps options.

- For Error Handling, choose the option that best suits your purpose.
- For the Always Run Step option, select the checkbox to ensure this step will always run.

Storing and Retrieving Properties in a Job

When a job is running, it is often necessary to share data between steps. Because the data being shared is specific to the running procedure instance, the job, the data is more appropriately stored at the job level rather than the procedure level. Use the techniques illustrated in this tutorial to safely pass data between steps in a job.

This tutorial shows you how to create, set, and retrieve properties stored on a job.

To view an example procedure for this tutorial, go to the automation platform UI > **Projects** > **EC-Tutorials-<version>**, and click the procedure name.

An explanation of what you see:

This procedure implements two steps.

- Write data to a property attached to the running job:
"Step 1 - Create a property attached to the job" calls ectool to create a property name `property1`.
 - The path to the property being set is `/jobs/${myJob}/property1`.
 - `${myJob}` is a shortcut to the running job name and can be used while the job is running.
If the job number was 1000, the path `/jobs/${myJob}/property1` might expand to `/jobs/job-1000/property1`.
- Read data from the property created in Step 1:
"Step 2 - Read the property attached to the job by Step 1" - this step could read step values stored on a job, running as part of the job, or could be executed using a simple property expansion.
 - In this case, `property1` was set in Step 1 and is referenced in Step 2 using the `${property1}` notation.
 - Two other ways to reference a property (shown by example in the Action column):
 - Using `${myJob/property1}`
 - Using inline JavaScript with the notation `[/javascript myJob.property1]`

Click **Run** to run this sample procedure and see the resulting job on the Job Details page.

- After the job completes, click the icon in the Log column for Step 2 to view three lines each showing the data (the property value) that was stored in the property that was retrieved.

Implementation

If you need to retrieve properties, adapt these step concepts to your procedures.

- For help getting started, click on a step name in the tutorial procedure to go to the corresponding Edit Step page.
- You can reuse the text supplied in the Command(s) box, changing the "property1" name and other values to those more meaningful to your project/procedure.

Note: Any changes you make within this tutorial will not be saved when you upgrade ElectricFlow.

Related information

[Procedure - create new or edit existing procedure](#) - Help topic

[Step - create new or edit existing step](#) - Help topic

Working with Properties Stored in a Procedure

Using ElectricFlow properties is one of the keys to robust yet flexible process automation. Configuration and historical data persistence are common tasks in process automation.

This tutorial shows common ways to read and write properties on a procedure.

Note: For this tutorial, familiarity with creating procedures and steps is assumed. See the "Related Information" links below if you would like to review these topics.

To view an example procedure for this tutorial, go to the automation platform UI > **Projects** > **EC-Tutorials-<version>**, and click the procedure name.

An explanation of what you see—this procedure implements the following functionality:

- This procedure implements five steps. Each step name is a description of its action.
 - Click on a step name to go to the corresponding Edit Step page to see how the step was created. This page also contains text in the Description text box to describe what you see in the Action column on the Procedure Details page. The full text of what appears in the Action column is provided in the Command text box.
 - In the Custom Procedure Properties table, notice two properties were created: `property1` and `property2`
- There are numerous ways to work with stored properties. ***Each of the 5 steps described below illustrates a different method.***
 - Read data from `property1` using the *property substitution* notation (syntax):
The property value is read in this step using the property substitution reference `$/myProcedure/property1`.
 - Read data from `property1` using inline JavaScript:
The property name `property1` is read from the procedure level using this JavaScript notation: `$/javascript myProcedure.property1`.
 - Perform a calculation using data from `property 2` and inline JavaScript:
In this example, the value of `property2` is multiplied by 5 using this JavaScript notation: `$/javascript myProcedure.property2 * 5`.
 - Read data from a `property1` using the ElectricFlow Perl API:
Perl code is used to call the ElectricFlow API to retrieve the value of `property1`.
The `getProperty` API returns an object so the property value must be retrieved after the `getProperty` call, using the `findnodes` method for the returned object.

```
#read the property
my $property1node = $ec->getProperty("/myProcedure/property1");
#retrieve the property value from the object returned to $property1node
my $property1 = $property1node->findnodes("//value")->string_value();
```

- Read data from `property1` using `ectool`:
`ectool` code is used to retrieve the value of `property1`:

```
ectool getProperty "/projects/$[/myProject/projectName]/procedures/Working
with properties stored in a procedure/property1"
```

Click **Run** to run this sample procedure and see the results on the Job Details page.

On the Job Details page, click the icon in the Log column to see the value read from the property.

Implementation

If you need to retrieve properties stored in a procedure, adapt any one of these step concepts to your procedures.

You can reuse the text supplied in the Command(s) box (from any Edit Step page in this tutorial), changing names and other values to those more meaningful to your project/procedure.

Note: Any changes you make within this tutorial will not be saved when you upgrade ElectricFlow.

Related Information

[Procedure - create new or edit existing procedure](#) - Help topic

[Step - create new or edit existing step](#) - Help topic

[Properties](#) - Help topic