



SERENA®

RELEASE AUTOMATION

Plug-ins Guide

Serena Proprietary and Confidential Information

Copyright © 2013 Serena Software, Inc. All rights reserved.

This document, as well as the software described in it, is furnished under license and may be used or copied only in accordance with the terms of such license. Except as permitted by such license, no part of this publication may be reproduced, photocopied, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written permission of Serena. Any reproduction of such software product user documentation, regardless of whether the documentation is reproduced in whole or in part, must be accompanied by this copyright statement in its entirety, without modification. This document contains proprietary and confidential information, and no reproduction or dissemination of any information contained herein is allowed without the express permission of Serena Software.

The content of this document is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Serena. Serena assumes no responsibility or liability for any errors or inaccuracies that may appear in this document.

License and copyright information for 3rd party software included in this release can be found on the product's news page at <http://support.serena.com/ProductNews/default.aspx> and may also be found as part of the software download available at <http://www.support.serena.com>.

Trademarks

Serena, TeamTrack, StarTool, PVCS, Comparex, Dimensions, Prototype Composer, Mariner and ChangeMan are registered trademarks of Serena Software, Inc. The Serena logo, Version Manager and Mover are trademarks of Serena Software, Inc. All other products or company names are used for identification purposes only, and may be trademarks of their respective owners.

U.S. Government Rights

Any Software product acquired by Licensee under this Agreement for or on behalf of the U.S. Government, its agencies and instrumentalities is "commercial software" as defined by the FAR. Use, duplication, and disclosure by the U.S. Government is subject to the restrictions set forth in the license under which the Software was acquired. The manufacturer is Serena Software, Inc., 1850 Gateway Drive, 4th Floor, San Mateo, CA 94404.

Part number: Serena ALM Product version: 4.5.1

Publication date: 2013-07-30

Table of Contents

Part 1: Introduction	11
Chapter 1: Serena Release Automation Plug-ins	13
What are Plug-ins?	13
Standard Plug-ins.....	15
Chapter 2: List of Available Plug-ins	17
Chapter 3: Examples of Plug-in Usage	21
Expanded Process Example with Plug-ins	21
Part 2: Application Server Management Plug-ins	23
Chapter 4: Apache HTTP Server Plug-in	25
Creating an Apache Start Step	25
Creating an Apache Stop Step	25
Chapter 5: Apache Tomcat Plug-in.....	27
Chapter 6: GlassFish Plug-in	29
Chapter 7: IBM WebSphere Application Server Plug-in	31
Creating a WebSphere Check Application is Not Running Step	32
Creating a WebSphere Check Application is Running on Server or Cluster Step	34
Creating a WebSphere Check if Config Object Exists Step	35
Creating a WebSphere Check Initial Heap Size Step	36
Creating a WebSphere Check Maximum Heap Size Step	37
Creating a WebSphere Check Status Step.....	38
Creating a WebSphere Create Cluster step	40
Creating a WebSphere Create Cluster Member Step	40
Creating a WebSphere Create DataSource Step	41
Creating a WebSphere Create DataSource for Cluster Step	43
Creating a WebSphere Create JDBCProvider for Cluster Step	45
Creating a WebSphere Create JDBCProvider Step	46
Creating a WebSphere Create a SIBJMSConnectionFactory Step	47
Creating a WebSphere Create SIBJMSQueue Step	49

Creating a WebSphere Create SIBJMSTopic Step	50
Creating an WebSphere Execute wsadmin Script Step	51
Creating a WebSphere Export DataSource Properties from a Cluster Step	53
Creating an WebSphere Export JVMHeapSizes from Server Step	54
Creating a WebSphere Export Application Step	55
Creating a WebSphere Generate Plugin Step	56
Creating a WebSphere Import DataSource Properties into Cluster Step	58
Creating a WebSphere Import JVMHeapSizes for Cluster Step	59
Creating a WebSphere Install Application Step	60
Creating a WebSphere Install or Update Application Step.....	62
Creating a WebSphere Modify Application ClassLoaders Step	64
Creating a WebSphere Restart Application Step	65
Creating a WebSphere Restart Server Step	66
Creating a WebSphere Start Application Step	68
Creating a WebSphere Start Server Step	69
Creating a WebSphere Stop Application Step	70
Creating a WebSphere Stop Server Step	71
Creating a WebSphere Synchronize Nodes Step	73
Creating a WebSphere Uninstall Application Step	73
Creating a WebSphere Update J2EEResourceProperty on a Config Object Step	75
Creating a WebSphere Update Simple Attribute on Object Step	76
Creating a WebSphere Update Application Step	77
Creating a WebSphere Wait for Application Step.....	79
Creating a Websphere Wait for Server or Cluster Step	81
Chapter 8: IBM WebSphere MB Plug-in.....	83
Creating a WebSphere WMB Set Broker Properties Step	83
Creating a WebSphere WMB Set Execution Group Properties Step.....	84
Creating a WebSphere WMB Set Message Flows Property Step	85
Creating a WebSphere WMB Delete Flows Using RegEx Step	86

Creating a WebSphere WMB Deploy Step	87
Creating a WebSphere WMB Start Message Flows Step.....	88
Creating a WebSphere WMB Stop Message Flows Step	89
Chapter 9: IBM WebSphere MQ	91
Creating a WebSphere Create Queue Manager Step	91
Creating a WebSphere Manage Queue Depth Step	92
Creating a WebSphere Start Queue Manager Step	92
Creating a WebSphere Stop Queue Manager Step	93
Creating a WebSphere Define Queue Step	93
Creating a WebSphere Run MQ Script Step	94
Chapter 10: JBoss Plug-in	97
Creating a Deploy Application Step	98
Creating a Deploy JDBC Driver Step	98
Creating an Undeploy Application Step	99
Creating a Start JBoss Step	100
Creating a Stop JBoss Step	100
Creating an Add Data Source Step	101
Creating an Add JMS Queue Step	101
Creating an Add JMS Topic Step	102
Creating an Add JMS Connection Factory Step.....	103
Creating a Remove JMS Connection Factory Step	103
Creating a Create Server Group Step	104
Creating a Run Script Step	104
Creating a Check Deployment Status Step	105
Creating an Enable Application Step	106
Creating a Disable Application Step	106
Chapter 11: IIS AdminScripts Plug-in	107
Creating an AdsUtil Step	107
Creating an Update WebSiteProperties Step.....	108
Creating an Update VDirProperties Step	108

Creating a Set .Net Version Step	108
Chapter 12: IIS AppCmd Plug-in	111
Creating a Create Application Step	111
Creating a Delete Application Step	112
Creating a Create Site Step	112
Creating a Delete Site Step	113
Creating a Start Site Step	113
Creating a Stop Site Step	114
Creating a Check if Site Exists Step	114
Creating a Check if Site is Stopped Step	115
Creating a Check if Site is Running Step	115
Creating a Create Virtual Directory Step	116
Creating a Delete Virtual Directory Step	116
Creating a Create Application Pool Step.....	117
Creating a Delete Application Pool Step.....	117
Creating a Start Application Pool Step	118
Creating a Stop Application Pool Step	118
Creating a Recycle Application Pool Step	119
Creating an AppCmd Step	119
Chapter 13: Microsoft IIS MS-Deploy Plug-in	121
Creating a Synchronize Step.....	121
Creating an MSDeploy Step	123
Creating a Delete Step	124
Creating a Stop Application Step	125
Creating a Recycle Application Step	126
Chapter 14: Oracle WebLogic WLDeploy Plug-in	127
Part 3: Build Management Plug-ins	129
Chapter 15: Apache Ant Plug-in	131
Creating an Ant Step	131
Chapter 16: Apache Maven Plug-in	133

Chapter 17: MS Build Plug-in	135
Chapter 18: UrbanCode AnthillPro Plug-in	137
Part 4: Configuration Management and Repository Plug-ins	139
Chapter 19: CollabNet Teamforge Plug-in	141
Chapter 20: Servicenow Plug-in	143
Chapter 21: Microsoft Sharepoint Plug-in.....	145
Chapter 22: Serena PVCS VM Plug-in	147
Creating a PVCS Export Step	147
Chapter 23: Subversion Plug-in	149
Part 5: Database Management Plug-ins	151
Chapter 24: DBDeploy Plug-in.....	153
Creating a Run DbDeploy Step	153
Chapter 25: DBUpgrader Plug-in	155
Chapter 26: SQLCMD Plug-in	157
Chapter 27: SQL*Plus Plug-in	159
Creating a Run SQLPlus Script Step	159
Chapter 28: SQL-JDBC Plug-in	161
Part 6: Integration Plug-ins	163
Chapter 29: Informatica Plug-in	165
Chapter 30: Microsoft BizTalk Plug-in	167
Part 7: Network Management Plug-ins.....	169
Chapter 31: Citrix NetScaler Plug-in	171
Chapter 32: F5 BIG-IP Plug-in.....	173
Chapter 33: Nagios XI Plug-in.....	175
Part 8: Quality and Test Management Plug-ins	177
Chapter 34: Atlassian Jira Plug-in	179
Chapter 35: Deploy Tools Plug-in	181
Chapter 36: HP Quality Center Plug-in	183
Creating a Create Issue Step	183
Creating an Update Issues Step	184
Creating an Add Comments Step.....	186

Creating a Publish Issue Report Step	187
Creating a Run Test Set Step	188
Creating a Publish Test Set Report Step	188
Creating a Check Status Step	189
Creating a Query Defects Step	190
Chapter 37: QuickTest Pro Plug-in	193
Chapter 38: Rally Plug-in	195
Chapter 39: Selenium Plug-in	197
Part 9: Release Automation Management Plug-ins	199
Chapter 40: BMC ARA Plug-in	201
Chapter 41: Serena RA Application Plug-in	203
Chapter 42: Serena RA Component Plug-in	205
Chapter 43: Serena RA Configuration Management Plug-in	207
Chapter 44: Serena RA Environment Plug-in	209
Chapter 45: Serena RA Resource Plug-in	211
Chapter 46: Serena Release Automation System Plug-in.....	213
Creating a Serena Release Automation System Property Step	213
Chapter 47: Serena RA Version Plug-in	215
Chapter 48: Serena RA VFS Plug-in	217
Part 10: System Tools and Scripting Plug-ins	219
Chapter 49: 7zip Plug-in	221
Creating a 7zip Extract Archive Step	221
Chapter 50: CA AutoSys Plug-in	223
Chapter 51: FileUtils Plug-in	225
Chapter 52: Groovy Plug-in	227
Chapter 53: Microsoft Message Queuing (MSMQ) Plug-in	229
Chapter 54: Microsoft Software Installer (MSI) Plug-in	231
Chapter 55: Microsoft Windows Service Control Manager Plug-in	233
Chapter 56: Windows System Tools Plug-in	235
Chapter 57: Red Hat Package Manager (RPM) Plug-in	237

Creating an Install RPM Step	237
Creating an Uninstall RPM Step	237
Creating an Update RPM Step	237
Chapter 58: Shell Plug-in	239
Chapter 59: System Information Plug-in	241
Part 11: Virtual and Cloud Environment Management Plug-ins	243
Chapter 60: Amazon EC2 Plug-in	245
Creating an Amazon EC2 Launch Instance Step	245
Creating an Amazon EC2 Security Group Step	246
Amazon EC2 Start, Stop, and Terminate Instance Steps	247
Creating an Amazon EC2 Wait for Instance Step	248
Creating an Amazon EC2 Associate IPs Step	248
Creating an Amazon EC2 Register Instances with LoadBalance Step	249
Creating an Amazon EC2 Deregister Instances with LoadBalance Step	249
Creating an Amazon EC2 Get Public DNS Step	250
Chapter 61: Microsoft Windows Azure Plug-in	251
Chapter 62: VMware vSphere ESXi Plug-in	253
Chapter 63: VMware vCenter Plug-in	255
Chapter 64: VMware Workstation Plug-in	257
Part 12: Creating Your Own Plug-ins	259
Chapter 65: Plug-in Creation	261
Plug-in Creation Overview	261
The plugin.xml File	263
<header> Element	263
Plug-in Steps: <step-type> Element	264
Step Properties: <properties> Element	265
<command> Element	267
The <post-processing> Element	268
Upgrading Plug-ins	269
The info.xml File	270

Part 1: Introduction

This section contains the following information:

- [Chapter 1: Serena Release Automation Plug-ins \[page 13\]](#)
- [Chapter 2: List of Available Plug-ins \[page 17\]](#)
- [Chapter 3: Examples of Plug-in Usage \[page 21\]](#)

Chapter 1: Serena Release Automation Plug-ins

This guide lists and describes the plug-ins included with Serena Release Automation. When you are creating processes, use these plug-ins to configure and execute steps needed for your deployment, such as requests to application servers, build management tools, database and file management systems, and test automation tools.

All of the plug-ins include self-documentation. As you use plug-ins within the processes, click the ? icon beside the properties fields to see a short description of each property.

See the *Serena Release Automation User's Guide* for information on using plug-ins in processes and creating new plug-ins, including examples.



Note: If you require documentation for a plug-in beyond that provided in the Serena Release Automation guides, please check the Serena Support Knowledgebase and follow your usual Serena Support procedures.



Important: The plug-in steps specified in the Serena Release Automation processes often require pre-existing conditions, such as existing accounts, licenses, security, and login credentials. You should consult the integrating products' documentation for information on how to configure and use the integrating software.

What are Plug-ins?

Serena Release Automation plug-ins provide tools for creating component processes. Plug-ins consist of configurable *steps* which can be thought of as distinct pieces of automation. Plug-ins integrate many third-party tools into Serena Release Automation, such as application servers and software configuration management products.

By combining steps in the Serena Release Automation editor, you can create fully-automated deployment processes. For example, the Tomcat and WebSphere plug-ins provide steps that start and stop those servers, install and uninstall applications, as well as perform other tool-specific tasks.

Each plug-in step consists of a number of properties, a command that performs the function associated with the step, and post-processing instructions, which typically are used to ensure that expected results occurred. Step properties can serve many purposes, from providing information required by the step's command, to supplying some or all of the actual command itself.

When you create a process, you drag steps onto the editor's design area and define their properties as you go. Property values can be supplied when defining a component process or at run-time. The process flow is defined by drawing connections between steps.

In the following illustration, you can see a series of plug-in steps and the connections between them. For information about creating component processes and creating your own post-processing scripts, see the *Serena Release Automation User's Guide*.

Figure 1. Example Process



Plug-ins at Run-time

Component processes are run by agents installed in the target environment. For a process to run successfully, the agent must have access to all resources, tools, and files required by the plug-in steps used in the process. When installing an agent, ensure that:

- The agent running the process has the necessary user permissions to execute commands and access any required resources. This typically entails granting permissions if an external tool is installed as a different user; installing the agent as a service; or impersonating the appropriate user.
- Any external tools required by plug-in steps are installed in the target environment.

-
- The required minimum version of any external tool is installed.

Standard Plug-ins

Serena Release Automation includes a standard set of plug-ins that appear in the UI by default. You do not need to load these plug-ins unless you have removed them and need to re-add them, or you want to upgrade them to the latest version.

The standard plug-ins are listed and described in the following table.

Plug-in	Description
File Utils	Used to perform folder and file level tasks as part of a deployment process. Includes steps for deleting and creating directories and replacing tokens in a file.
Resource Properties Collector	Used for retrieving environment and resource properties from a resource.
Serena RA Application	Used for creating and managing Serena RA applications.
Serena RA Component	Used for creating and managing Serena RA components.
Serena RA Configuration Management	Used for downloading and uploading configuration templates from and to Serena RA.
Serena RA Environment	Used for creating and managing Serena RA environments.
Serena RA Resource	Used for managing Serena RA Resources.
Serena RA System	Used for managing Serena RA system properties and global settings.
Serena RA Version	Used for editing Component Versions in Serena RA.
Serena RA Versioned File Storage	Used to upload artifacts to a Serena RA (VFS) artifact repository.

Chapter 2: List of Available Plug-ins

The plug-ins provided with Serena Release Automation are listed here, divided into broad usage categories.



Note: New plug-ins are actively in development. If you don't see a plug-in that you need in this list, please search the Serena Support Knowledgebase and follow your usual Serena Support procedures.

Application Server Management	Build Management	Configuration Management and Repositories
<ul style="list-style-type: none">• Apache HTTP• Apache Tomcat• GlassFish• IBM WebSphere<ul style="list-style-type: none">▪ WebSphere Application Server▪ WebSphere Message Broker - CMP▪ WebSphere MQ• JBoss• Microsoft IIS<ul style="list-style-type: none">▪ AdminScripts▪ AppCmd▪ MS-Deploy• Oracle WebLogic Server WLDeploy	<ul style="list-style-type: none">• Apache Ant• Apache Maven• Microsoft Build Engine (MSBuild)• UrbanCode AnthillPro	<ul style="list-style-type: none">• CollabNet TeamForge• ServiceNow Change Management• Microsoft SharePoint• Serena PVCS• Subversion

Database Management	Integration	Network Management
<ul style="list-style-type: none"> • DBDeploy • DBUpgrader • Microsoft SQL Server SQLCMD • Oracle SQL*Plus • SQL-JDBC 	<ul style="list-style-type: none"> • Informatica (Data Integration) • Microsoft BizTalk Server (Application Integration) 	<ul style="list-style-type: none"> • Citrix NetScaler • F5 BIG-IP • Nagios XI
Quality and Test Management	Release Automation Management	System Tools and Scripting
<ul style="list-style-type: none"> • Atlassian JIRA • Deploy Tools • HP Quality Center • QuickTest Pro • Rally Platform • Selenium 	<ul style="list-style-type: none"> • BMC Application Release Automation (ARA) • Serena Release Automation <ul style="list-style-type: none"> ▪ Application ▪ Component ▪ Configuration Management ▪ Environment ▪ Resource ▪ System ▪ Version ▪ Versioned File Storage (VFS) 	<ul style="list-style-type: none"> • 7zip • Autosys (Unix/Linux) • FileUtils (Unix/Linux) • Groovy • Microsoft <ul style="list-style-type: none"> ▪ Message Queuing (MSMQ) ▪ Software Installer (MSI) ▪ Windows Service Control Manager ▪ Windows System Tools • Red Hat Package Manager (RPM) • Shell (Windows/Unix/Linux) • System Information (Unix/Linux)

Virtual and Cloud Environment Management	
---	--

- | | |
|--|--|
| <ul style="list-style-type: none">• Amazon EC2• Microsoft Azure• VMware vSphere ESXi• VMware vCenter• VMware Workstation | |
|--|--|

Chapter 3: Examples of Plug-in Usage

Following are a few high-level examples of how plug-ins are used in component processes. As you design your processes, your deployment needs will determine what plug-in steps you need to use.

Example 1: Mapping AnthillPro Components According to Type

You might take artifacts from a source such as an AnthillPro project and map the ones that get deployed to an HTTP server into one component, those that get deployed to a J2EE container to another, and those that get deployed to a database to yet another.

Example 2: Single-component Deployment with Two Scheduled Processes

A single-component deployment might consist of two processes: the first moves component files to a server on Friday night (a lengthy operation), while the second deploys the files Saturday morning. You would likely want to use one or more plug-in steps to determine if the first process created all of the prerequisite conditions before allowing the second scheduled process to begin.

Example 3: Deploying to a WebSphere Application Server

Deploying a J2EE EAR file to WebSphere Application Server typically consists of the following operations:

1. Transfer the EAR file to the target machine.
2. Stop the WebSphere server instance.
3. Invoke wsAdmin with deployment properties.
4. Start the WebSphere instance.
5. Verify that the deployment succeeded by accessing a specified URL.



Note: The WebSphere plug-in provides a configurable process step for each operation.

A frequently-used component process can be saved as a template and applied later to new components.

Expanded Process Example with Plug-ins

As you begin automating your deployment processes, you will likely find that you need one or more plug-ins to prepare the target environments as shown in the following example.

The fictitious Qlarius QSocial application includes an APP portion (Java-based) and a WEB portion (files). In this example, two processes are used to deploy the two QSocial application components, or artifacts.

The APP component process uses the following steps:

1. Delete Files and Directories (from System Utilities > FileUtils plugins)

2. Download Artifacts (from Repositories-Artifact > SerenRA plugins)
3. Start (from Application Server > Apache plugins)
4. Undeploy Application (from Java > Tomcat plugins)
5. Deploy Application (from Java > Tomcat plugins)

The WEB component process uses the following steps:

1. Delete Files and Directories (from System Utilities > FileUtils plugins)
2. Download Artifacts (from Repositories-Artifact > SerenRA plugins)
3. Delete Files and Directories (from System Utilities > FileUtils plugins)
4. Copy Directories (from System Utilities > FileUtils plugins)

Part 2: Application Server Management Plug-ins

This section contains the following information:

- [Chapter 4: Apache HTTP Server Plug-in \[page 25\]](#)
- [Chapter 5: Apache Tomcat Plug-in \[page 27\]](#)
- [Chapter 6: GlassFish Plug-in \[page 29\]](#)
- [Chapter 7: IBM WebSphere Application Server Plug-in \[page 31\]](#)
- [Chapter 8: IBM WebSphere MB Plug-in \[page 83\]](#)
- [Chapter 9: IBM WebSphere MQ \[page 91\]](#)
- [Chapter 10: JBoss Plug-in \[page 97\]](#)
- [Chapter 11: IIS AdminScripts Plug-in \[page 107\]](#)
- [Chapter 12: IIS AppCmd Plug-in \[page 111\]](#)
- [Chapter 13: Microsoft IIS MS-Deploy Plug-in \[page 121\]](#)
- [Chapter 14: Oracle WebLogic WLDeploy Plug-in \[page 127\]](#)

Chapter 4: Apache HTTP Server Plug-in

Apache HTTP Server is a public domain open source Web server for UNIX and Windows NT operating systems. This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Start
- Stop

For information about Apache HTTP Server, see <http://httpd.apache.org/>.

Creating an Apache Start Step

This step starts Apache HTTP server.

Required properties: None.

Optional properties:

Property Name	Description
apachectl executable path	The full path (including the executable) to the apachectl script, if not on the path. This is used only on *nix systems.
Windows Service name	The name of the Windows server used to control the Apache HTTP server. This is only used on Windows systems.

Creating an Apache Stop Step

This step stops Apache HTTP server.

Required properties: None.

Optional properties:

Property Name	Description
apachectl executable path	The full path (including the executable) to the apachectl script, if not on the path. This is used only on *nix systems.
Windows Service name	The name of the Windows server used to control the Apache HTTP server. This is only used on Windows systems.

Chapter 5: Apache Tomcat Plug-in

Apache Tomcat is an open source Web application server and servlet container. The Tomcat plug-in may be used during deployments to execute Tomcat runbook automation and deploy or un-deploy Tomcat applications.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Deploy Application
- Undeploy Application
- Start Application
- Stop Application
- Start Tomcat
- Stop Tomcat
- Check if Tomcat is alive

For information about Apache Tomcat, see <http://tomcat.apache.org/tomcat-7.0-doc/index.html>.

Chapter 6: GlassFish Plug-in

GlassFish is an open source application server project started by Sun Microsystems for the Java EE platform; it is now sponsored by Oracle Corporation. This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Deploy Application
- Undeploy Application
- Redeploy Application
- Enable Application
- Disable Application
- Start Domain
- Stop Domain
- Execute asadmin

For information about GlassFish, see <https://glassfish.java.net/documentation.html>.

Chapter 7: IBM WebSphere Application Server Plug-in

The IBM WebSphere Application Server, also known as WAS, is an Application Server built using J2EE, XML, and Web Services. WAS is supported on Windows, AIX, and Linux, as well as other operating systems.

IBM WebSphere Application Server provides a Java application server runtime environment and it can be used to connect website users to servlets. It allows users to create and manage sophisticated business websites.

The WebSphere Application Server plug-in contains the following steps for you to add to your processes in Serena Release Automation:

- Install Application
- Install or Update Application
- Export Application
- Update Application
- Uninstall Application
- Start Application
- Stop Application
- Restart Application
- Start Server
- Stop Server
- Restart Server
- Wait for Application
- Execute wsadmin Script
- Check Status
- Wait for server or cluster
- Check Application is not running
- Check Application is running on server or cluster
- Generate Plugin
- Synchronize nodes

- Modify Application ClassLoaders
- Create Cluster
- Create Cluster Member
- Check Initial Heap Size
- Check Maximum Heap Size
- Export JVMHeapSizes from Server
- Import JVMHeapSizes for Cluster
- Export DataSource Properties from Cluster
- Import DataSource Properties into Cluster
- Create JDBCProvider For Cluster
- Create JDBCProvider
- Create DataSource For Cluster
- Create DataSource
- Check If Config Object Exists
- Create SIBJMSQueue
- Create SIBJMSTopic
- Create SIBJMSConnectionFactory
- Update Simple Attribute on Object
- Update J2EEResourceProperty on Object

For information about WAS, see <http://www-01.ibm.com/software/websphere/>.

Creating a WebSphere Check Application is Not Running Step

This step is used to ensure an application is not running.

Required properties:

Property Name	Description
Name	Enter the name of your Check Application is Not Running step.
Application Name	Set the name of the application that you want to check.

Property Name	Description
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE

Optional properties:

Property Name	Description
Cell Name	The name of the cell to be administered. The default value is: <code>\${p:resource/websphere.cell}</code> .
Node Name	The name of the node to be administered. The default value is: <code>\${p:resource/websphere.node}</code> .
Server Name	The name of the server to be administered.
Cluster Name	The name of the cluster to be administered.
Command Path	The directory location of the wsadmin command-line executable. For example: <pre> /opt/IBM/WebSphere/AppServer/bin/ </pre> The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.
Additional Command Line Arguments	New line separated list of additional command line arguments to pass to wsadmin. These will be appended as the last arguments of the command line.

Creating a WebSphere Check Application is Running on Server or Cluster Step

Use this step to check to make sure an application is running on a server or cluster.

Required properties:

Property Name	Description
Name	Enter the name for your Check Application is Not Running step.
Application Name	Set the name of the application that you want to check.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE

Optional properties:


Property Name	Description
Cell Name	The name of the cell to be administered. The default value is: <code>\${p:resource/websphere.cell}</code> .
Node Name	The name of the node to be administered. The default value is: <code>\${p:resource/websphere.node}</code> .
Server Name	The name of the server to be administered.
Cluster Name	The name of the cluster to be administered.
Command Path	The directory location of the wsadmin command-line executable. For example: <pre> /opt/IBM/WebSphere/AppServer/bin/ </pre> The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .

Property Name	Description
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.
Additional Command Line Arguments	New line separated list of additional command line arguments to pass to wsadmin. These will be appended as the last arguments of the command line.

Creating a WebSphere Check if Config Object Exists Step

Use this step to check if a config object exists in the WebSphere Configuration by Containment Path.

Required properties:

Property Name	Description
Name	Enter a name for the Check if Config Object Exists step.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE
Object Containment Path	Enter the containment path to the object to check for. For example: /Cell:cellName/ServerCluster:clusterName/ JDBCProvider:providerName/.  Note: The path must end with a JDBCProvider type and a "/".

Optional properties:

Property Name	Description
Command Path	The directory location of the wsadmin command-line executable. For example: /opt/IBM/WebSphere/AppServer/bin/ The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.

Creating a WebSphere Check Initial Heap Size Step

Use this step to check the initial heap size.

Required properties:

Property Name	Description
Name	Enter the name for your Check Initial Heap Size step.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE
Server Name	Enter the name of the server to check.
Minimum Size (MB)	Enter the minimum expected heap size in MBs.

Property Name	Description
Maximum Size (MB)	Enter the maximum expected heap size in MBs.

Optional properties:

Property Name	Description
Command Path	The directory location of the wsadmin command-line executable. For example: <pre>/opt/IBM/WebSphere/AppServer/bin/</pre> The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.

Creating a WebSphere Check Maximum Heap Size Step

Use this step to check the maximum heap size.

Required properties:

Property Name	Description
Name	Enter the name for your Check Maximum Heap Size step.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE

Property Name	Description
Server Name	Enter the name of the server to check.
Minimum Size (MB)	Enter the minimum expected heap size in MBs.
Maximum Size (MB)	Enter the maximum expected heap size in MBs.

Optional properties:

Property Name	Description
Command Path	<p>The directory location of the wsadmin command-line executable. For example:</p> <pre style="text-align: center;">/opt/IBM/WebSphere/AppServer/bin/</pre> <p>The default value is: <code>\${p:resource/websphere.commandPath}</code>.</p>
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.

Creating a WebSphere Check Status Step

This step is used to check the status of a WebSphere server or cluster.

Required properties:

Property Name	Description
Name	Enter the name for your Check Status step.

Property Name	Description
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE

Optional properties:

Property Name	Description
Cell Name	The name of the cell to be administered. The default value is: <code>\${p:resource/websphere.cell}</code> .
Node Name	The name of the node to be administered. The default value is: <code>\${p:resource/websphere.node}</code> .
Server Name	The name of the server to be administered.
Cluster Name	The name of the cluster to be administered.
Command Path	The directory location of the wsadmin command-line executable. For example: <pre> /opt/IBM/WebSphere/AppServer/bin/ </pre> The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.
Additional Command Line Arguments	New line separated list of additional command line arguments to pass to wsadmin. These will be appended as the last arguments of the command line.

Creating a WebSphere Create Cluster step

This step is used to create a new WebSphere cluster.

Required properties:

Property Name	Description
Name	Enter the name for your Create Cluster step.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE
Cluster Name	Enter the name of the new cluster that you want to create.

Optional properties:

Property Name	Description
Command Path	The directory location of the wsadmin command-line executable. For example: <pre> /opt/IBM/WebSphere/AppServer/bin/ </pre> The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.

Creating a WebSphere Create Cluster Member Step

This step is used to create a new WebSphere cluster member.

Required properties:

Property Name	Description
Name	Enter the name for your Create Cluster Member step.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none">• SOAP• RMI• NONE
Cluster Name	Enter the name of the new cluster member that you want to create.
Node Name	Enter the name of the node for the new cluster member.


Optional properties:

Property Name	Description
Command Path	The directory location of the wsadmin command-line executable. For example: <code>/opt/IBM/WebSphere/AppServer/bin/</code> The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.

Creating a WebSphere Create DataSource Step

Use this step to create a DataSource on a given JDBCProvider.

Required properties:

Property Name	Description
Name	Enter the name for your Create DataSource step.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE
JDBCProvider Location	Enter the containment path to the JDBCProvider. For example: /Cell:cellName/ServerCluster:clusterName/ JDBCProvider:providerName/.  Note: The path must end with a JDBCProvider type and a "/".
DataSource Name	Enter the name of the datasource you want to create.
DataStore Helper ClassName	Enter the ClassName of the DataStore Helper.
DB Name	Enter the database name for the datasource.
JNDI Name	Enter the JNDI name for the datasource.
Auth Alias	Enter the auth alias to create or use for the datasource.
Alias Username	Enter the username for the Auth Alias if it does not already exist.
Alias Password	Enter the password for the Auth Alias if it does not already exist.
Description	Enter a description for the datasource.

Optional properties:

Property Name	Description
Command Path	The directory location of the wsadmin command-line executable. For example: <code>/opt/IBM/WebSphere/AppServer/bin/</code> The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.

Creating a WebSphere Create DataSource for Cluster Step

This step is used to create a DataSource on a WebSphere cluster.

Required properties:

Property Name	Description
Name	Enter the name for your Create DataSource for Cluster step.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none">• SOAP• RMI• NONE
Cluster Name	Enter the name of the cluster that hosts the JDBCProvider.
JDBCProvider Name	Enter the name of the JDBCProvider.

Property Name	Description
DataSource Name	Enter the name of the datasource you want to create.
DataStore Helper ClassName	Enter the ClassName of the DataStore Helper.
DB Name	Enter the database name for the datasource.
JNDI Name	Enter the JNDI name for the datasource.
Driver Type	Enter the type of driver that is used by the datasource.
Server Name	Enter the name of the server to which the datasource connects.
Port Number	Enter the server's port number.
Auth Alias	Enter the auth alias to create or use for the datasource.
Alias Username	Enter the username for the Auth Alias if it does not already exist.
Alias Password	Enter the password for the Auth Alias if it does not already exist.
Description	Enter a description for the datasource.

Optional properties:

Property Name	Description
Command Path	<p>The directory location of the wsadmin command-line executable. For example:</p> <pre style="text-align: center;">/opt/IBM/WebSphere/AppServer/bin/</pre> <p>The default value is: <code>\${p:resource/websphere.commandPath}</code>.</p>
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.

Creating a WebSphere Create JDBCProvider for Cluster Step

Use this step to create a JDBCProvider on a cluster.

Required properties:

Property Name	Description
Name	Enter the name for your Create JDBCProvider for Cluster step.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none">• SOAP• RMI• NONE
Cluster Name	Enter the name of the cluster that hosts the JDBCProvider.
JDBCProvider Name	Enter the name of the JDBCProvider.
DataSource Name	Enter the name of the datasource you want to create.
DataStore Helper ClassName	Enter the ClassName of the DataStore Helper.
DB Name	Enter the database name for the datasource.
JNDI Name	Enter the JNDI name for the datasource.
Driver Type	Enter the type of driver that is used by the datasource.
Server Name	Enter the name of the server to which the datasource connects.
Port Number	Enter the server's port number.
Auth Alias	Enter the auth alias to create or use for the datasource.
Alias Username	Enter the username for the Auth Alias if it does not already exist.
Alias Password	Enter the password for the Auth Alias if it does not already exist.
Description	Enter a description for the datasource.

Optional properties:

Property Name	Description
Command Path	The directory location of the wsadmin command-line executable. For example: /opt/IBM/WebSphere/AppServer/bin/ The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.

Creating a WebSphere Create JDBCProvider Step

Use this step to create a JDBCProvider with specified scope.

Required properties:

Property Name	Description
Name	Enter the name for your Create JDBCProvider step.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE
Scope	Enter the scope at which to create the JDBCProvider in the format: <code>type=name</code> . The type can be a Cell, Node, Server, Application, or Cluster, and name is a specific instance of the cell, node, server, application, or cluster that you are using.
DB Type	Enter the database type of the JDBCProvider you want to create.

Property Name	Description
Provider Type	Enter the provider type of the JDBCProvider you want to create.
Implementation Type	Enter the implementation type of the JDBCProvider.
Description	Enter a description for the JDBCProvider.
Class Path	Enter the class path of the JDBCProvider.
Native Path	Enter the native path of the JDBCProvider.


Optional properties:

Property Name	Description
Command Path	<p>The directory location of the wsadmin command-line executable. For example:</p> <pre style="text-align: center;">/opt/IBM/WebSphere/AppServer/bin/</pre> <p>The default value is: <code>\${p:resource/websphere.commandPath}</code>.</p>
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.

Creating a WebSphere Create a SIBJMSConnectionFactory Step

Use this step to create a SIBJMSConnectionFactory on a give scope.

Required properties:

Property Name	Description
Name	Enter the name for your Create SIMJBSCONNECTIONFACTORY step.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE
Scope	Enter the scope, which is the containment path to the parent on which to create this SIBJMSSCONNECTIONFACTORY. For example: /Cell:cellName/ServerCluster:clusterName/.  Note: The path must end with a "/".
SIBJMSSCONNECTIONFACTORY Name	Enter a name for the SIBJMSSCONNECTIONFACTORY you want to create.
JNDI Name	Enter a JNDI name for the factory you want to create.
Bus Name	Enter the name of the SIBus to associate with this Connection Factory.
Type	Enter the type of Connection Factory to create. Use <code>Queue</code> or <code>Topic</code> .

Optional properties:


Property Name	Description
Command Path	The directory location of the wsadmin command-line executable. For example: /opt/IBM/WebSphere/AppServer/bin/ The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .

Property Name	Description
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.

Creating a WebSphere Create SIBJMSQueue Step

This step will create a SIBJMSQueue on a given scope.

Required properties:

Property Name	Description
Name	Enter the name for your Create SIBJMSQueue step.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE
Scope	Enter the scope, which is the containment path to the parent on which to create this SIBJMSQueue. For example: <code>/Cell:cellName/ServerCluster:clusterName/</code> . <p> Note: The path must end with a "/".</p>
SIBJMSQueue Name	Enter a SIBJMSQueue name for the queue you want to create.
JNDI Name	Enter a JNDI name for the queue you want to create.
Queue Name	Enter the queue name of the service integration bus destination to which this queue maps.

Optional properties:


Property Name	Description
Command Path	The directory location of the wsadmin command-line executable. For example: /opt/IBM/WebSphere/AppServer/bin/ The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.

Creating a WebSphere Create SIBJMSTopic Step

This step will create a SIBJMSTopic on a given scope.

Required properties:

Property Name	Description
Name	Enter the name for your Create SIMJBSTopic step.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE

Property Name	Description
Scope	<p>Enter the scope, which is the containment path to the parent on which to create this SIBJMSTopic. For example: <code>/Cell:cellName/ServerCluster:clusterName/</code>.</p> <p> Note: The path must end with a "/".</p>
SIBJMSTopic Name	Enter a SIBJMSTopic name for the topic you want to create.
JNDI Name	Enter a JNDI name for the topic you want to create.

Optional properties:

Property Name	Description
Command Path	<p>The directory location of the wsadmin command-line executable. For example:</p> <p><code>/opt/IBM/WebSphere/AppServer/bin/</code></p> <p>The default value is: <code>\${p:resource/websphere.commandPath}</code>.</p>
User Name	The user name to be used to connect to the WebSphere Node. The default value is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.

Creating an WebSphere Execute wsadmin Script Step

This step is used to execute a Jython or JACL script through wsadmin.

Required properties:

Property Name	Description
Name	Enter the name for your Execute wsadmin Script step.
Script Path	Enter the script path to call wsadmin.
Language	Enter the language for the script. Possible values are: Jython or JACL.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE

Optional properties:

Property Name	Description
Cell Name	The name of the cell to be administered. The default value is: <code>\${p:resource/websphere.cell}</code> .
Node Name	The name of the node to be administered. The default value is: <code>\${p:resource/websphere.node}</code> .
Server Name	The name of the server to be administered.
Cluster Name	The name of the cluster to be administered.
Command Path	The directory location of the wsadmin command-line executable. For example: <code>/opt/IBM/WebSphere/AppServer/bin/</code> The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.

Property Name	Description
Host	Hostname of the server to connect to.
Port	The port to connect to.
Additional Command Line Arguments	New line separated list of additional command line arguments to pass to wsadmin. These will be appended as the last arguments of the command line.

Creating a WebSphere Export DataSource Properties from a Cluster Step

This step is used to export the datasource properties from a cluster.

Required properties:

Property Name	Description
Name	Enter the name for your Export DataSource Properties from a Cluster step.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE
Cluster Name	Enter the cluster name that hosts the JDBCProvider.
JDBCProvider Name	Enter the name of the JDBCProvider for the datasource.
DataSource Name	Enter the name of the datasource from which to get properties.

Optional properties:

Property Name	Description
Command Path	The directory location of the wsadmin command-line executable. For example: /opt/IBM/WebSphere/AppServer/bin/ The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.

Creating an WebSphere Export JVMHeapSizes from Server Step

Use this step to export the JVM heap size values from a WebSphere server.

Required properties:

Property Name	Description
Name	Enter the name for your Export JVMHeapSizes from Server step.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE
Server Name	Enter the name of the server from which to export the heap size.

Optional properties:

Property Name	Description
Command Path	The directory location of the wsadmin command-line executable. For example: <code>/opt/IBM/WebSphere/AppServer/bin/</code> The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.

Creating a WebSphere Export Application Step

This step is used to export an application from the WebSphere server.

Required properties:

Property Name	Description
Name	Enter the name of your Export Application step.
Application Name	Set the name of the application that you want to export. This is the name used by WebSphere for reference.
File Path	Set the path of the file to store. This must be expressed as an absolute path.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none">• SOAP• RMI• NONE

Optional properties:

Property Name	Description
Cell Name	The name of the cell to be administered. The default value is: <code>\${p:resource/websphere.cell}</code> .
Node Name	The name of the node to be administered. The default value is: <code>\${p:resource/websphere.node}</code> .
Server Name	The name of the server to be administered.
Cluster Name	The name of the cluster to be administered.
Command Path	The directory location of the wsadmin command-line executable. For example: <code>/opt/IBM/WebSphere/AppServer/bin/</code> The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.
Additional Command Line Arguments	New line separated list of additional command line arguments to pass to wsadmin. These will be appended as the last arguments of the command line.

Creating a WebSphere Generate Plugin Step

Use this step to generate a WebSphere plugin for WebServers to use.

Required properties:

Property Name	Description
Name	Enter the name of your Generate Plugin step.

Property Name	Description
Application Server Root Directory	Enter the path for the application server root directory.
Configuration Repo Root Directory	Enter the root directory of the configuration repo were the plugin should be created
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE

Optional properties:

Property Name	Description
Cell Name	The name of the cell to be administered. The default value is: <code>\${p:resource/websphere.cell}</code> .
Node Name	The name of the node to be administered. The default value is: <code>\${p:resource/websphere.node}</code> .
Server Name	The name of the server to be administered.
Cluster Name	The name of the cluster to be administered.
Command Path	The directory location of the wsadmin command-line executable. For example: <pre> /opt/IBM/WebSphere/AppServer/bin/ </pre> The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.

Property Name	Description
Port	The port to connect to.
Additional Command Line Arguments	New line separated list of additional command line arguments to pass to wsadmin. These will be appended as the last arguments of the command line.

Creating a WebSphere Import DataSource Properties into Cluster Step

Use this step to import the datasource properties into a cluster.

Required properties:

Property Name	Description
Name	Enter the name for your Import Datasource Properties into a Cluster step.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE
Cluster Name	Enter the cluster name for the JDBCProvider.
JDBCProvider Name	Enter the JDBCProvider Name for the datasource.
DataSource Name	Enter the name of the datasource from which to get properties.
Connection Timeout	Enter the connection timeout to use.
Statement Cache Size	Enter the statement cache size to use.

Optional properties:

Property Name	Description
Command Path	The directory location of the wsadmin command-line executable. For example: <code>/opt/IBM/WebSphere/AppServer/bin/</code> The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.

Creating a WebSphere Import JVMHeapSizes for Cluster Step

Use this step to import the JVM heap size values for all servers in a cluster.

Required properties:

Property Name	Description
Name	Enter the name for your Import JVMHeapSizes for Cluster step.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none">• SOAP• RMI• NONE
Cluster Name	Enter the name of the cluster to be updated.
Initial Heap Size	Enter the initial heap size to use for all servers in the cluster.

Property Name	Description
Maximum Heap Size	Enter the maximum heap size to use for all servers in the cluster.

Optional properties:

Property Name	Description
Command Path	The directory location of the wsadmin command-line executable. For example: <code>/opt/IBM/WebSphere/AppServer/bin/</code> The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.

Creating a WebSphere Install Application Step

This step is used to install an application to the WebSphere server or cluster. It is well suited to installing an application to WebSphere for the first time.

Required properties:

Property Name	Description
Name	Enter the name for your Install or Install Application step.
Context Root	Set the context root path for this application. Default is "/".
Application Source	Set the location of the application to install to WebSphere.
Application Name	The name of the application you want to install. This is used for WebSphere reference.

Property Name	Description
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE
Server Name	Enter the name of the server to check.
Minimum Size (MB)	Enter the minimum expected heap size in MBs.
Maximum Size (MB)	Enter the maximum expected heap size in MBs.

Optional properties:

Property Name	Description
Application Path	The path for the application to be installed on the server.
Options String	The string of options to be concatenated onto the install command.
Cell Name	The name of the cell to be administered. The default value is: <code>\${p:resource/websphere.cell}</code> .
Node Name	The name of the node to be administered. The default value is: <code>\${p:resource/websphere.node}</code> .
Server Name	The name of the server to be administered.
Cluster Name	The name of the cluster to be administered.
Command Path	The directory location of the wsadmin command-line executable. For example: <pre> /opt/IBM/WebSphere/AppServer/bin/ </pre> The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .

Property Name	Description
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.
Additional Command Line Arguments	New line separated list of additional command line arguments to pass to wsadmin. These will be appended as the last arguments of the command line.

Creating a WebSphere Install or Update Application Step

This step is used to install or update an existing application to the WebSphere server or cluster. It is suited to reinstalling an application over an existing application on the WebSphere server.

Required properties:

Property Name	Description
Name	Enter the name for your Install or Update Application step.
Context Root	Set the context root path for this application. Default is "/".
Application Source	Set the location of the application to install to WebSphere.
Application Name	The name of the application you want to install. This is used for WebSphere reference.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE
Server Name	Enter the name of the server to check.

Property Name	Description
Minimum Size (MB)	Enter the minimum expected heap size in MBs.
Maximum Size (MB)	Enter the maximum expected heap size in MBs.

Optional properties:

Property Name	Description
Application Path	The path for the application to be installed on the server.
Options String	The string of options to be concatenated onto the install command.
Cell Name	The name of the cell to be administered. The default value is: <code>\${p:resource/websphere.cell}</code> .
Node Name	The name of the node to be administered. The default value is: <code>\${p:resource/websphere.node}</code> .
Server Name	The name of the server to be administered.
Cluster Name	The name of the cluster to be administered.
Command Path	The directory location of the wsadmin command-line executable. For example: <code>/opt/IBM/WebSphere/AppServer/bin/</code> The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.

Property Name	Description
Additional Command Line Arguments	New line separated list of additional command line arguments to pass to wsadmin. These will be appended as the last arguments of the command line.

Creating a WebSphere Modify Application ClassLoaders Step

Use this step if you need to modify the classloaders for an application and its webmodules.

Required properties:

Property Name	Description
Name	Enter the name for your Modify Application ClassLoaders step.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE

Optional properties:

Property Name	Description
Command Path	The directory location of the wsadmin command-line executable. For example: <pre> /opt/IBM/WebSphere/AppServer/bin/ </pre> The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.

Property Name	Description
Host	Hostname of the server to connect to.
Port	The port to connect to.
Application Name	The name of the application for which to update classloaders.
Application ClassLoader	PARENT_FIRST or PARENT_LAST. The Classloader mode to use for the overall application. The default value is PARENT_FIRST.
WebModule ClassLoader	PARENT_FIRST or PARENT_LAST. The Classloader mode to use for the webmodules in the given application. The default value is PARENT_FIRST.
Additional CommandLine Arguments	New line separated list of additional commandline arguments to pass to wsadmin. These will be appended as the last arguments of the command line.

Creating a WebSphere Restart Application Step

This step is used to restart an application on the WebSphere server or cluster.

Required properties:

Property Name	Description
Name	Enter the name of your Restart Application step.
Application Name	Set the name of the application that you want to restart. This is the name used for WebSphere reference.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE

Optional properties:

Property Name	Description
Cell Name	The name of the cell to be administered. The default value is: <code>\${p:resource/websphere.cell}</code> .
Node Name	The name of the node to be administered. The default value is: <code>\${p:resource/websphere.node}</code> .
Server Name	The name of the server to be administered.
Cluster Name	The name of the cluster to be administered.
Command Path	The directory location of the wsadmin command-line executable. For example: <code>/opt/IBM/WebSphere/AppServer/bin/</code> The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.
Additional Command Line Arguments	New line separated list of additional command line arguments to pass to wsadmin. These will be appended as the last arguments of the command line.

Creating a WebSphere Restart Server Step

This step is used to restart a WebSphere server or cluster.

Required properties:

Property Name	Description
Name	Enter the name of your Restart Server step.

Property Name	Description
Connection Type	<p>Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are:</p> <ul style="list-style-type: none"> • SOAP • RMI • NONE

Optional properties:

Property Name	Description
Cell Name	The name of the cell to be administered. The default value is: <code>\${p:resource/websphere.cell}</code> .
Node Name	The name of the node to be administered. The default value is: <code>\${p:resource/websphere.node}</code> .
Server Name	The name of the server to be administered.
Cluster Name	The name of the cluster to be administered.
Command Path	<p>The directory location of the wsadmin command-line executable. For example:</p> <pre style="text-align: center;">/opt/IBM/WebSphere/AppServer/bin/</pre> <p>The default value is: <code>\${p:resource/websphere.commandPath}</code>.</p>
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.
Additional Command Line Arguments	New line separated list of additional command line arguments to pass to wsadmin. These will be appended as the last arguments of the command line.

Creating a WebSphere Start Application Step

This step is used to start an application on the WebSphere server or cluster.

Required properties:

Property Name	Description
Name	Enter the name of your Start Application step.
Application Name	Set the name of the application that you want to start. This is the name used for WebSphere reference.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE

Optional properties:

Property Name	Description
Cell Name	The name of the cell to be administered. The default value is: <code>\${p:resource/websphere.cell}</code> .
Node Name	The name of the node to be administered. The default value is: <code>\${p:resource/websphere.node}</code> .
Server Name	The name of the server to be administered.
Cluster Name	The name of the cluster to be administered.
Command Path	The directory location of the wsadmin command-line executable. For example: <pre> /opt/IBM/WebSphere/AppServer/bin/ </pre> The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.

Property Name	Description
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.
Additional Command Line Arguments	New line separated list of additional command line arguments to pass to wsadmin. These will be appended as the last arguments of the command line.

Creating a WebSphere Start Server Step

This step is used to start a WebSphere server or cluster.

Required properties:

Property Name	Description
Name	Enter the name of your Start Server step.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE

Optional properties:

Property Name	Description
Cell Name	The name of the cell to be administered. The default value is: <code>\${p:resource/websphere.cell}</code> .
Node Name	The name of the node to be administered. The default value is: <code>\${p:resource/websphere.node}</code> .
Server Name	The name of the server to be administered.
Cluster Name	The name of the cluster to be administered.

Property Name	Description
Command Path	The directory location of the wsadmin command-line executable. For example: /opt/IBM/WebSphere/AppServer/bin/ The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.
Additional Command Line Arguments	New line separated list of additional command line arguments to pass to wsadmin. These will be appended as the last arguments of the command line.

Creating a WebSphere Stop Application Step

This step is used to stop an application on the WebSphere server or cluster.

Required properties:

Property Name	Description
Name	Enter the name of your Stop Application step.
Application Name	Set the name of the application that you want to stop. This is the name used for WebSphere reference.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE

Optional properties:

Property Name	Description
Cell Name	The name of the cell to be administered. The default value is: <code>\${p:resource/websphere.cell}</code> .
Node Name	The name of the node to be administered. The default value is: <code>\${p:resource/websphere.node}</code> .
Server Name	The name of the server to be administered.
Cluster Name	The name of the cluster to be administered.
Command Path	The directory location of the wsadmin command-line executable. For example: <code>/opt/IBM/WebSphere/AppServer/bin/</code> The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.
Additional Command Line Arguments	New line separated list of additional command line arguments to pass to wsadmin. These will be appended as the last arguments of the command line.

Creating a WebSphere Stop Server Step

This step is used to stop a WebSphere server or cluster.

Required properties:

Property Name	Description
Name	Enter the name of your Stop Server step.

Property Name	Description
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE

Optional properties:

Property Name	Description
Cell Name	The name of the cell to be administered. The default value is: <code>\${p:resource/websphere.cell}</code> .
Node Name	The name of the node to be administered. The default value is: <code>\${p:resource/websphere.node}</code> .
Server Name	The name of the server to be administered.
Cluster Name	The name of the cluster to be administered.
Command Path	The directory location of the wsadmin command-line executable. For example: <pre> /opt/IBM/WebSphere/AppServer/bin/ </pre> The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.
Additional Command Line Arguments	New line separated list of additional command line arguments to pass to wsadmin. These will be appended as the last arguments of the command line.

Creating a WebSphere Synchronize Nodes Step

Use this step to synchronize all nodes in the WebSphere cell.

Required properties:

Property Name	Description
Name	Enter the name for your Synchronize Nodes step.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none">• SOAP• RMI• NONE

Optional properties:

Property Name	Description
Command Path	The directory location of the wsadmin command-line executable. For example: <code>/opt/IBM/WebSphere/AppServer/bin/</code> The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.

Creating a WebSphere Uninstall Application Step

This step is used to uninstall an application from the WebSphere server or cluster.

Required properties:

Property Name	Description
Name	Enter the name of your Uninstall Application step.
Application Name	Set the name of the application that will be uninstalled. This is the name used for WebSphere reference.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE

Optional properties:


Property Name	Description
Cell Name	The name of the cell to be administered. The default value is: <code>\${p:resource/websphere.cell}</code> .
Node Name	The name of the node to be administered. The default value is: <code>\${p:resource/websphere.node}</code> .
Server Name	The name of the server to be administered.
Cluster Name	The name of the cluster to be administered.
Command Path	The directory location of the wsadmin command-line executable. For example: <pre> /opt/IBM/WebSphere/AppServer/bin/ </pre> The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.

Property Name	Description
Host	Hostname of the server to connect to.
Port	The port to connect to.
Additional Command Line Arguments	New line separated list of additional command line arguments to pass to wsadmin. These will be appended as the last arguments of the command line.

Creating a WebSphere Update J2EEResourceProperty on a Config Object Step

This step is used to create or update a J2EEResourceProperty on a config object.

Required properties:

Property Name	Description
Name	Enter the name for your Update J2EEResourceProperty on a Config Object step.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE
Object Containment Path	Enter the path to the object on which to update a property. For example: <code>/Cell:cellName/ServerCluster:clusterName/</code> .  Note: The path must end with a "/".
Property Name	Enter the name of the property you want to update.
Value	Enter the value for the property.
Type	Enter the type of the property.

Optional properties:


Property Name	Description
Command Path	The directory location of the wsadmin command-line executable. For example: /opt/IBM/WebSphere/AppServer/bin/ The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.

Creating a WebSphere Update Simple Attribute on Object Step

Use this step to update a simple attribute on a config object.

Required properties:

Property Name	Description
Name	Enter the name for your Update Simple Attribute on Object step.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE

Property Name	Description
Object Containment Path	Enter the path to the object on which to update an attribute. For example: <code>/Cell:cellName/ServerCluster:clusterName/</code> .  Note: The path must end with a "/".
Attribute Name	Enter the name of the attribute you want to update.
Value	Enter the value for the attribute.

Optional properties:

Property Name	Description
Command Path	The directory location of the wsadmin command-line executable. For example: <code>/opt/IBM/WebSphere/AppServer/bin/</code> The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.

Creating a WebSphere Update Application Step

This step is used to update an application on the WebSphere server or cluster.

Required properties:

Property Name	Description
Name	Enter the name of your Install Application step.

Property Name	Description
Application Name	Set the name of the application that is being updated. This is the name used for WebSphere reference.
Content Type	Set the content type for the update. Possible values are: <ul style="list-style-type: none"> • Application • File • Module File • Partial Application
Operation	Set the operation to be performed during this update. (If this is an application update or a partial application update, you must enter <code>update</code> .) Possible values are: <ul style="list-style-type: none"> • Update • Add • Delete • Add or Update
Content Path	Set the location of the contents that will update the application.
Connection Type	Select the Connection Type to use with <code>wsadmin</code> . The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE

Optional properties:

Property Name	Description
Content URI	If the content type is File, this should be the location of the file to be updated, relative to the EAR root.
Options String	The string of options to be concatenated onto the update command.

Property Name	Description
Cell Name	The name of the cell to be administered. The default value is: <code>\${p:resource/websphere.cell}</code> .
Node Name	The name of the node to be administered. The default value is: <code>\${p:resource/websphere.node}</code> .
Server Name	The name of the server to be administered.
Cluster Name	The name of the cluster to be administered.
Command Path	The directory location of the wsadmin command-line executable. For example: <pre style="text-align: center;">/opt/IBM/WebSphere/AppServer/bin/</pre> The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.
Additional Command Line Arguments	New line separated list of additional command line arguments to pass to wsadmin. These will be appended as the last arguments of the command line.

Creating a WebSphere Wait for Application Step

This step is used to create a wait step while a WebSphere server or cluster becomes ready.

Required properties:

Property Name	Description
Name	Enter the name of your Wait for Application step.

Property Name	Description
Application Name	Set the name of the application to wait for.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE

Optional properties:

Property Name	Description
Cell Name	The name of the cell to be administered. The default value is: <code>\${p:resource/websphere.cell}</code> .
Node Name	The name of the node to be administered. The default value is: <code>\${p:resource/websphere.node}</code> .
Server Name	The name of the server to be administered.
Cluster Name	The name of the cluster to be administered.
Command Path	The directory location of the wsadmin command-line executable. For example: <pre> /opt/IBM/WebSphere/AppServer/bin/ </pre> The default value is: <code>\${p:resource/websphere.commandPath}</code> .
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.

Property Name	Description
Additional Command Line Arguments	New line separated list of additional command line arguments to pass to wsadmin. These will be appended as the last arguments of the command line.

Creating a Websphere Wait for Server or Cluster Step

Use this step to build in a wait period for a WebSphere server or cluster to be started.

Required properties:

Property Name	Description
Name	Enter the name of your Wait for Server or Cluster step.
Connection Type	Select the Connection Type to use with wsadmin. The default value is "SOAP". Possible values are: <ul style="list-style-type: none"> • SOAP • RMI • NONE

Optional properties:

Property Name	Description
Cell Name	The name of the cell to be administered. The default value is: <code>\${p:resource/websphere.cell}</code> .
Node Name	The name of the node to be administered. The default value is: <code>\${p:resource/websphere.node}</code> .
Server Name	The name of the server to be administered.
Cluster Name	The name of the cluster to be administered.
Command Path	The directory location of the wsadmin command-line executable. For example: <pre> /opt/IBM/WebSphere/AppServer/bin/ </pre> The default value is: <code>\${p:resource/websphere.commandPath}</code> .

Property Name	Description
User Name	The user name to be used to connect to the WebSphere Node. The default values is <code>\${p:resource/websphere.user}</code> .
Password	The password to be used to connect to the WebSphere Node.
Password Script	If you wish to use a script or property lookups for your password, leave the Password field blank and enter it here.
Host	Hostname of the server to connect to.
Port	The port to connect to.
Additional Command Line Arguments	New line separated list of additional command line arguments to pass to wsadmin. These will be appended as the last arguments of the command line.

Chapter 8: IBM WebSphere MB Plug-in

The IBM WebSphere Message Broker (WebSphere MB) API is also known as the Configuration Manager Proxy (CMP) or CMP API. The CMP API consists of a Java implementation and allows applications to control brokers and their resources.

WebSphere MB can translate messages between two devices using different protocols for communication. This allows heterogeneous applications to communicate without having to know specific implementation details about the application with which they are trying to communicate.

The WebSphere MB plug-in enables you to deploy broker archives and start and stop message flows using the CMP API.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- WMB Set Broker Properties
- WMB Set Execution Group Properties
- WMB Set Message Flows Property
- WMB Delete Flows Using RegEx
- WMB Deploy
- WMB Start Message Flows
- WMB Stop Message Flows

For information about WebSphere MB, see <http://www-03.ibm.com/software/products/us/en/integration-bus>.

Creating a WebSphere WMB Set Broker Properties Step

This step sets runtime properties for the broker.

Required properties:

Property Name	Description
Properties	A newline-separated list of properties that you want to set, provided in <code>name=value</code> format.
IP	The IP address of the target server.
Port	The port of the target server.

Property Name	Description
Queue Manager	The WebSphere Queue Manager that the broker is using.

Optional properties:

Property Name	Description
Username	Specify an alternate username for authentication against WMQ in case the step is running as an invalid user.
Trace File	Used to specify the trace logging file. If no value is supplied, then trace logging is disabled.
Jar Path	The full path that contains both the <code>ConfigManagerProxy.jar</code> and the <code>com.ibm.mq.jar</code> . For example: <code>C:\Program Files\IBM\MQSI\7.0\classes\ConfigManagerProxy.jar:C:\Program Files\IBM\WebSphere MQ\java lib\com.ibm.mq.jar</code>

Creating a WebSphere WMB Set Execution Group Properties Step

This step sets runtime properties for an execution group.

Required properties:

Property Name	Description
Execution Group	The name of the Execution Group to deploy to.
Properties	A newline-separated list of properties that you want to set, provided in <code>name=value</code> format.
IP	The IP address of the target server.
Port	The port of the target server.
Queue Manager	The WebSphere Queue Manager that the broker is using.

Optional properties:

Property Name	Description
Username	Specify an alternate username for authentication against WMQ in case the step is running as an invalid user.
Trace File	Used to specify the trace logging file. If no value is supplied, then trace logging is disabled.
Jar Path	The full path that contains both the <code>ConfigManagerProxy.jar</code> and the <code>com.ibm.mq.jar</code> . For example: <code>C:\Program Files\IBM\MQSI\7.0\classes\ConfigManagerProxy.jar:C:\Program Files\IBM\WebSphere MQ\java lib\com.ibm.mq.jar</code>

Creating a WebSphere WMB Set Message Flows Property Step

This step sets runtime properties on a list of message flow.

Required properties:

Property Name	Description
Execution Group	The name of the Execution Group to deploy to.
IP	The IP address of the target server.
Port	The port of the target server.
Queue Manager	The WebSphere Queue Manager that the broker is using.

Optional properties:

Property Name	Description
Directory Offset	The path under the current working directory where the step should execute. Absolute paths are not allowed.
Message Flows	A newline-separated list of message flows.
Properties	A newline-separated list of properties that you want to set, provided in <code>name=value</code> format.

Property Name	Description
Property File	The name and path to a property file that contains the properties. This can be used instead of explicitly providing all the properties in the Properties field and the flows in the Message Flows field. You must specify the property file in the same format as the <code>mqsipplybaroverride</code> command.
Username	Specify an alternate username for authentication against WMQ in case the step is running as an invalid user.
Trace File	Used to specify the trace logging file. If no value is supplied, then trace logging is disabled.
Jar Path	The full path that contains both the <code>ConfigManagerProxy.jar</code> and the <code>com.ibm.mq.jar</code> . For example: <code>C:\Program Files\IBM\MQSI\7.0\classes\ConfigManagerProxy.jar:C:\Program Files\IBM\WebSphere MQ\java lib\com.ibm.mq.jar</code>

Creating a WebSphere WMB Delete Flows Using RegEx Step

This step deletes WMB flows that have been deployed using the BAR file that matches a provided regular expression. This is useful for deploying into shared execution groups.

Required properties:

Property Name	Description
RegEx	A Java compliant regular expression that is used to locate the correct BAR file of deployed flows by matching its filename.
Execution Group	The name of the Execution Group to deploy to.
IP	The IP address of the target server.
Port	The port of the target server.
Queue Manager	The WebSphere Queue Manager that the broker is using.

Optional properties:

Property Name	Description
Timeout	The amount of time to wait for each BAR file deploy result to come return (in milliseconds). The default value is 60000.
Username	Specify an alternate username for authentication against WMQ in case the step is running as an invalid user.
Trace File	Used to specify the trace logging file. If no value is supplied, then trace logging is disabled.
Jar Path	The full path that contains both the <code>ConfigManagerProxy.jar</code> and the <code>com.ibm.mq.jar</code> . For example: <code>C:\Program Files\IBM\MQSI\7.0\classes\ConfigManagerProxy.jar</code> ; <code>C:\Program Files\IBM\WebSphere MQ\java lib\com.ibm.mq.jar</code>

Creating a WebSphere WMB Deploy Step

This step deploys a broker archive.

Required properties:

Property Name	Description
BAR File Names	A comma-separated list of broker archive files to deploy.
Execution Group	The name of the Execution Group to deploy to.
IP	The IP address of the target server.
Port	The port of the target server.
Queue Manager	The WebSphere Queue Manager that the broker is using.

Optional properties:

Property Name	Description
Value for Full Deploy	If selected, the execution group is emptied before deploy.
Timeout	The amount of time to wait for each BAR file deploy result to come return (in milliseconds). The default value is 60000.

Property Name	Description
Value for Start/ Stop Message Flows	If selected, all message flows in the execution group are stopped before deployment and started after deployment.
Username	Specify an alternate username for authentication against WMQ in case the step is running as an invalid user.
Trace File	Used to specify the trace logging file. If no value is supplied, then trace logging is disabled.
Jar Path	The full path that contains both the <code>ConfigManagerProxy.jar</code> and the <code>com.ibm.mq.jar</code> . For example: <code>C:\Program Files\IBM\MQSI\7.0\classes\ConfigManagerProxy.jar:C:\Program Files\IBM\WebSphere MQ\java lib\com.ibm.mq.jar</code>

Creating a WebSphere WMB Start Message Flows Step

This step starts message flows that are stopped.

Required properties:

Property Name	Description
Execution Group	The name of the execution group.
IP	The IP address of the target server.
Port	The port of the target server.
Queue Manager	The WebSphere Queue Manager that the broker is using.

Optional properties:

Property Name	Description
Message Flows	A newline-separated list of message flow names to start.
Username	Specify an alternate username for authentication against WMQ in case the step is running as an invalid user.

Property Name	Description
Trace File	Used to specify the trace logging file. If no value is supplied, then trace logging is disabled.
Jar Path	The full path that contains both the <code>ConfigManagerProxy.jar</code> and the <code>com.ibm.mq.jar</code> . For example: <code>C:\Program Files\IBM\MQSI\7.0\classes\ConfigManagerProxy.jar:C:\Program Files\IBM\WebSphere MQ\java lib\com.ibm.mq.jar</code>

Creating a WebSphere WMB Stop Message Flows Step

This step stops message flows that are running.

Required properties:

Property Name	Description
Execution Group	The name of the execution group.
IP	The IP address of the target server.
Port	The port of the target server.
Queue Manager	The WebSphere Queue Manager that the broker is using.

Optional properties:

Property Name	Description
Message Flows	A newline-separated list of message flow names to stop.
Username	Specify an alternate username for authentication against WMQ in case the step is running as an invalid user.
Trace File	Used to specify the trace logging file. If no value is supplied, then trace logging is disabled.
Jar Path	The full path that contains both the <code>ConfigManagerProxy.jar</code> and the <code>com.ibm.mq.jar</code> . For example: <code>C:\Program Files\IBM\MQSI\7.0\classes\ConfigManagerProxy.jar:C:\Program Files\IBM\WebSphere MQ\java lib\com.ibm.mq.jar</code>

Chapter 9: IBM WebSphere MQ

The IBM WebSphere Message Queuing enables multiple heterogeneous applications to communicate across a network of dissimilar components such as processors, operating systems, subsystems, and communication protocols, using a consistent API so that the message delivery process is decoupled from the actual application. Any type of data can be transported as messages.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Create Queue Manager
- Manage Queue Depth
- Start Queue Manager
- Stop Queue Manager
- Define Queue
- Run MQ Script

For information about WebSphere MQ, see <http://www-01.ibm.com/software/websphere/>.

Creating a WebSphere Create Queue Manager Step

This step runs the executable to create a Queue Manager.

Required properties:

Property Name	Description
Queue Name	The name of the queue manager you want to create.
Command Directory	Directory of the WebSphere MQ commands. For example: C:\Program Files (x86)\IBM\WebSphere MQ\bin.

Optional properties:

Property Name	Description
Queue Description	Descriptive text for the Queue Manager that is being created. The maximum description length is 64 characters.

Property Name	Description
Additional Arguments	Any additional arguments for the Create Queue Manager command.

Creating a WebSphere Manage Queue Depth Step

This step manages the queue depth.

Required properties:

Property Name	Description
Queue Manager Name	The name of the queue manager you want to edit.
Queue Name	The name of the queue you want to edit.
Command Directory	Directory of the WebSphere MQ commands. For example: <code>C:\Program Files (x86)\IBM\WebSphere MQ\bin.</code>

Optional properties:

Property Name	Description
High Depth Level	The percentage of a Queue depth that constitutes "high".
Low Depth Level	The percentage of a Queue depth that constitutes "low".
Max Depth Level	The maximum number of messages in the queue.
Additional Arguments	Any additional arguments for the Manage Queue Depth command.

Creating a WebSphere Start Queue Manager Step

This step starts a queue manager.

Required properties:

Property Name	Description
Queue Manager Name	The name of the queue manager you want to start.
Command Directory	Directory of the WebSphere MQ commands. For example: C:\Program Files (x86)\IBM\WebSphere MQ\bin.

Optional properties:

Property Name	Description
Additional Arguments	Any additional arguments for the Create Queue Manager command.



Note: The queue manager must be created before it can be started.

Creating a WebSphere Stop Queue Manager Step

This step stops a queue manager.

Required properties:

Property Name	Description
Queue Manager Name	The name of the queue manager you want to stop.
Command Directory	Directory of the WebSphere MQ commands. For example: C:\Program Files (x86)\IBM\WebSphere MQ\bin.

Optional properties:

Property Name	Description
Additional Arguments	Any additional arguments for the Create Queue Manager command.

Creating a WebSphere Define Queue Step

This step defines a queue.

Required properties:

Property Name	Description
Queue Manager Name	The name of the queue manager that you want to use to define a queue.
Queue Name	The name of the queue that you want to define.
Command Directory	Directory of the WebSphere MQ commands. For example: C:\Program Files (x86)\IBM\WebSphere MQ\bin.

Optional properties:

Property Name	Description
Additional Arguments	Any additional arguments for the Create Queue Manager command.



Note: The queue manager must be created before a queue can be created.

Creating a WebSphere Run MQ Script Step

This step runs a custom MQ script that consists of MQ Control commands.

Required properties:

Property Name	Description
Queue Manager Name	The name of the queue manager on which to run the script.
Script Data	The script to be run. Enter the actual commands that the script must run. For example: <code>DISPLAY QUEUE ('My Queue')</code> .
Command Directory	Directory of the WebSphere MQ commands. For example: C:\Program Files (x86)\IBM\WebSphere MQ\bin.

Optional properties:

Property Name	Description
Additional Arguments	Any additional arguments for the Create Queue Manager command.

Chapter 10: JBoss Plug-in

JavaBeans Open Source Software (JBoss) is an open-source application server that implements Java EE.

The JBoss plug-ins automate a number of operations for a JBoss server. There are JBoss plug-ins for JBoss 6 and earlier and JBoss 7. Load the plug-in that supports the version of JBoss you are working with.

The JBoss 7 plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Deploy Application
- Deploy JDBC Driver
- Undeploy Application
- Start JBoss
- Stop JBoss
- Add Data Source
- Add JMS Queue
- Add JMS Topic
- Add JMS Connection Factory
- Remove JMS Connection Factory
- Create Server Group
- Create Server
- Run Script
- Check Deployment Status
- Enable Application
- Disable Application

The JBoss 6 plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Check Status
- Start and wait
- Stop and wait

For information about JBoss, see <http://www.jboss.org/jbossas>.

Creating a Deploy Application Step

This step deploys an application to JBoss.

Required properties:

Property Name	Description
JBoss CLI Path	The path to the JBoss Command Line Interface executable. For example: C:\wildfly-8.0.0.Alpha1\wildfly-8.0.0.Alpha1\bin.
Application Name	The name of the application to be deployed (including its file extension). If a source is not provided, the named application will be enabled instead.

Optional properties:

Property Name	Description
Application/JAR path	The source file of the application to deploy. If a source is not provided, a name must be provided instead.
Value for Deploy To All Server Groups	If selected, the step will deploy to all servers. This is only applicable in Domain Mode.
Deploy To Server Groups	A comma-separated list of the servers to deploy to. This is only applicable in Domain Mode.
Controller Name	The controller name of the JBoss instance to connect to.

Creating a Deploy JDBC Driver Step

This step deploys a JDBC driver to JBoss.

Required properties:

Property Name	Description
DBC Driver Path	The path to the JDBC Driver JAR.
JBoss CLI Path	The path to the JBoss Command Line Interface executable. For example: C:\wildfly-8.0.0.Alpha1\wildfly-8.0.0.Alpha1\bin.

Optional properties:

Property Name	Description
Driver Name	The name of the driver. If a name is not provided, the file name (including .jar) is used as the name.
Value for Deploy To All Server Groups	If selected, the step will deploy to all servers. This is only applicable in Domain Mode.
Deploy To Server Groups	A comma-separated list of the servers to deploy to. This is only applicable in Domain Mode.

Creating an Undeploy Application Step

This step undeploys an application in JBoss.

Required properties:

Property Name	Description
Application Name	The name of the file to be undeployed (including its file extension).
JBoss CLI Path	The path to the JBoss Command Line Interface executable. For example: C:\wildfly-8.0.0.Alpha1\wildfly-8.0.0.Alpha1\bin.

Optional properties:

Property Name	Description
Value for Keep Content	If selected, the content of the JBoss server will be kept. The application will be deactivated, but nothing will be deleted.
Value for Undeploy From All Server Groups	If selected, this step will undeploy from all relevant servers. This is only applicable in Domain Mode.
Undeploy From Server Groups	A comma-separated list of the servers to undeploy from. This is only applicable in Domain Mode.
Controller Name	The controller name of the JBoss instance to connect to.



Note: The **Keep Content** checkbox is not compatible with this step.

Creating a Start JBoss Step

This step starts JBoss in either Standalone or Domain mode.

Required properties:

Property Name	Description
Startup Mode	Determines the mode in which JBoss should start. Values include: <ul style="list-style-type: none"> • Standalone • Domain
JBoss Startup Path	The path to the JBoss startup executable. For example: C:\wildfly-8.0.0.Alpha1\wildfly-8.0.0.Alpha1\bin.
Server Start Wait Timeout	The amount of time (in milliseconds) to wait for the server to start before declaring timeout. The default value is 10000.
Host Name	The host name of the server to be started.
Port Number	The port number of the server to be started.

Optional properties:

Property Name	Description
Server Configuration	Enter a file name to start the server according to a specific configuration.

Creating a Stop JBoss Step

This step stops JBoss in either Standalone or Domain mode.

Required properties:

Property Name	Description
JBoss Startup Path	The path to the JBoss startup executable. For example: C:\wildfly-8.0.0.Alpha1\wildfly-8.0.0.Alpha1\bin.

Property Name	Description
Server Mode	Specifies the server mode. Values include: <ul style="list-style-type: none"> • Standalone • Domain

Optional properties:

Property Name	Description
Domain Mode Host Name	The name of the host. This is only required when the server is in Domain Mode. The default value is <code>master</code> .

Creating an Add Data Source Step

This step stops JBoss in either Standalone or Domain mode.

Required properties:

Property Name	Description
JBoss Startup Path	The path to the JBoss startup executable. For example: C:\wildfly-8.0.0.Alpha1\wildfly-8.0.0.Alpha1\bin.
Server Mode	Specifies the server mode. Values include: <ul style="list-style-type: none"> • Standalone • Domain

Optional properties:

Property Name	Description
Domain Mode Host Name	The name of the host. This is only required when the server is in Domain Mode. The default value is <code>master</code> .

Creating an Add JMS Queue Step

This step adds a JMS Queue to JBoss.

Required properties:

Property Name	Description
JBoss Startup Path	The path to the JBoss startup executable. For example: C:\wildfly-8.0.0.Alpha1\wildfly-8.0.0.Alpha1\bin.
Queue Name	The name of the queue that you want to create.
Entries	A comma-separated list of JNDI names that the queue will be bound to.

Optional properties:

Property Name	Description
Profile	The profile to add the queue to. This is required in domain mode.
Controller Name	The controller name of the JBoss instance to connect to.

Creating an Add JMS Topic Step

This step adds a JMS topic to JBoss.

Required properties:

Property Name	Description
JBoss Startup Path	The path to the JBoss startup executable. For example: C:\wildfly-8.0.0.Alpha1\wildfly-8.0.0.Alpha1\bin.
Topic Name	The name of the topic that you want to create.
Entries	A comma-separated list of JNDI names that the topic will be bound to.

Optional properties:

Property Name	Description
Profile	The profile to add the topic to. This is required in domain mode.
Controller Name	The controller name of the JBoss instance to connect to.

Creating an Add JMS Connection Factory Step

This step adds a JMS topic to JBoss.

Required properties:

Property Name	Description
JBoss Startup Path	The path to the JBoss startup executable. For example: C:\wildfly-8.0.0.Alpha1\wildfly-8.0.0.Alpha1\bin.
Topic Name	The name of the topic that you want to create.
Entries	A comma-separated list of JNDI names that the topic will be bound to.

Optional properties:

Property Name	Description
Profile	The profile to add the topic to. This is required in domain mode.
Controller Name	The controller name of the JBoss instance to connect to.

Creating a Remove JMS Connection Factory Step

This step removes a JMS connection factory from JBoss.

Required properties:

Property Name	Description
JBoss Startup Path	The path to the JBoss startup executable. For example: C:\wildfly-8.0.0.Alpha1\wildfly-8.0.0.Alpha1\bin.
Factory Name	The name of the factory that you want to remove.

Optional properties:

Property Name	Description
Profile	The profile to remove the factory from. This is required in domain mode.

Property Name	Description
Controller Name	The controller name of the JBoss instance to connect to.

Creating a Create Server Group Step

This step creates a server group on a JBoss domain instance.

Required properties:

Property Name	Description
JBoss CLI Path	The path to the JBoss Command Line Interface executable. For example: C:\wildfly-8.0.0.Alpha1\wildfly-8.0.0.Alpha1\bin.
Host Name	The name of the server host.
Server Name	The name to give the server to be created.
Group Name	The server group to add this server to.
Socket Offset	The socket offset for this server.

Optional properties:

Property Name	Description
Value for Auto-Start	If selected, auto-start will be set to <code>true</code> .
Controller Name	The controller name of the JBoss instance to connect to.

Creating a Run Script Step

This step runs a custom script on JBoss using the CLI.

Required properties:

Property Name	Description
JBoss CLI Path	The path to the JBoss Command Line Interface executable. For example: C:\wildfly-8.0.0.Alpha1\wildfly-8.0.0.Alpha1\bin.
Script Data	The script to be executed.

Optional properties:

Property Name	Description
Controller Name	The controller name of the JBoss instance to connect to.



Note: For information on available commands, run this step with the following script: `help --commands`.

Creating a Check Deployment Status Step

This step gets the status information on a deployment.

Required properties:

Property Name	Description
JBoss CLI Path	The path to the JBoss Command Line Interface executable. For example: C:\wildfly-8.0.0.Alpha1\wildfly-8.0.0.Alpha1\bin.
Deployment Name	The name of the deployment to check (the filename and extension).

Optional properties:

Property Name	Description
Host Name	The name of the host for the deployment. This is required in domain mode.
Server Name	The name of the server for the deployment. This is required in domain mode.
Controller Name	The controller name of the JBoss instance to connect to.

Creating an Enable Application Step

This step enables an application that has been deployed to JBoss (standalone mode only).

Required properties:

Property Name	Description
JBoss CLI Path	The path to the JBoss Command Line Interface executable. For example: C:\wildfly-8.0.0.Alpha1\wildfly-8.0.0.Alpha1\bin.
Application Name	The name of the application to be deployed. If a source is not provided, the named application will be enabled instead.

Optional properties:

Property Name	Description
Controller Name	The controller name of the JBoss instance to connect to.

Creating a Disable Application Step

This step disables a standalone application that has been deployed to JBoss.

Required properties:

Property Name	Description
JBoss CLI Path	The path to the JBoss Command Line Interface executable. For example: C:\wildfly-8.0.0.Alpha1\wildfly-8.0.0.Alpha1\bin.
Application Name	The name of the application to be deployed. If a source is not provided, the named application will be enabled instead.

Optional properties:

Property Name	Description
Controller Name	The controller name of the JBoss instance to connect to.

Chapter 11: IIS AdminScripts Plug-in

Microsoft IIS is a Web server application packaged with Windows Server for hosting websites, services, and applications. The IIS AdminScripts plug-in enables you to automate IIS configuration changes during a deployment.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- AdsUtil
- Update WebSiteProperties
- Update VDirProperties
- Set .Net Version

For information about IIS, see <http://www.iis.net/home>.

Creating an AdsUtil Step

This step runs an arbitrary adsutil command.

Required properties:

Property Name	Description
Command	The command to run using adsutil.

Optional properties:


Property Name	Description
Path	The path of the node for which you are setting the property, combined with the name of the property you are setting. For example: <code>w3svc/1/ServerComment</code> .
Parameters	A newline-separated list of parameters to be set if needed for the command.
AdsUtil.vbs Path	The path to the <code>AdsUtil.vbs</code> executable including the file name. For example: <code>C:\InetPub\AdminScripts\adsutil.vbs</code> .
Cscript Path	The path to the <code>cscript.exe</code> executable including the file name if not on the path. For example: <code>C:\Windows\system32\cscript.exe</code> .

Creating an Update WebSiteProperties Step

This step sets properties on a Web site.

Required properties: None.

Optional properties:

Property Name	Description
Website	The Web site name.
Parameters	<p>A newline-separated list of parameters to be set using the syntax name=value.</p> <p> Note: "=" is not valid as part of the name</p>


Creating an Update VDirProperties Step

This step sets properties on a virtual directory.

Required properties:

Property Name	Description
VDir Offset	The offset from the Web site path to the virtual directory. For example, with a virtual directory named test , the value would be <code>/root/test</code> . The default virtual directory is <code>/root</code> .

Optional properties:

Property Name	Description
Website	The Web site name.
Parameters	<p>A newline-separated list of parameters to be set using the syntax name=value.</p> <p> Note: "=" is not valid as part of the name</p>

Creating a Set .Net Version Step

This step sets the .NET version for a Web site.

Required properties:

Property Name	Description
Version	<p>The version of .NET to set. The .NET framework for this version must be installed, and it must contain the <code>aspnet_regiis</code> executable in <code>%windir%\microsoft.net\framework\\${version}</code>. You must select one of the following values:</p> <ul style="list-style-type: none">• 2.0• 1.1

Optional properties:

Property Name	Description
Website	The Web site name.

Chapter 12: IIS AppCmd Plug-in

Microsoft IIS is a Web server application packaged with Windows Server for hosting websites, services, and applications.

`AppCmd.exe` is a command line tool for managing IIS 7 and above. The IIS AppCmd plug-in enables you to configure and query objects on a Web server and to return output in text or XML among other things.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Create Application
- Delete Application
- Create Site
- Delete Site
- Start Site
- Stop Site
- Check if Site Exists
- Check if Site is Stopped
- Check if Site is Running
- Create Virtual Directory
- Delete Virtual Directory
- Create Application Pool
- Delete Application Pool
- Start Application Pool
- Stop Application Pool
- Recycle Application Pool
- AppCmd

For information about IIS, see <http://www.iis.net/home>.

Creating a Create Application Step

This step creates an application in IIS.

Required properties:

Property Name	Description
Parent Site	The parent site the application should be created under. For example: <code>Default Web Site</code> .
Virtual Path	The virtual path under which the application should be created.

Optional properties:

Property Name	Description
Arguments	A newline-separated list of arguments to be added to the appcmd call. For example: <code>/arg1:val1\n/arg2:val2\n/arg3:val3...</code>
Command Path	The directory location of the AppCmd command-line executable. For example: <code>C:\Windows\system32\inetsrc\</code> .

Creating a Delete Application Step

This step deletes an application in IIS.

Required properties:

Property Name	Description
ID	The application path or URL of the application to delete. For example: <code>Default Web Site/Application</code> .

Optional properties:

Property Name	Description
Arguments	A newline-separated list of arguments to be added to the appcmd call. For example: <code>/arg1:val1\n/arg2:val2\n/arg3:val3...</code>
Command Path	The directory location of the AppCmd command-line executable. For example: <code>C:\Windows\system32\inetsrv\</code> .

Creating a Create Site Step

This step creates a new Web site in IIS.

Required properties:

Property Name	Description
Site Name	The name of the Web site to create. For example: <code>DefaultWebSite</code> .

Optional properties:

Property Name	Description
Arguments	A newline-separated list of arguments to be added to the appcmd call. For example: <code>/arg1:val1\n/arg2:val2\n/arg3:val3...</code>
Command Path	The directory location of the AppCmd command-line executable. For example: <code>C:\Windows\system32\inetsrv\.</code>

Creating a Delete Site Step

This step deletes a Web site in IIS.

Required properties:

Property Name	Description
Site Name	The name of the Web site to delete. For example: <code>DefaultWebSite</code> .

Optional properties:

Property Name	Description
Arguments	A newline-separated list of arguments to be added to the appcmd call. For example: <code>/arg1:val1\n/arg2:val2\n/arg3:val3...</code>
Command Path	The directory location of the AppCmd command-line executable. For example: <code>C:\Windows\system32\inetsrv\.</code>

Creating a Start Site Step

This step starts a Web site in IIS.

Required properties:

Property Name	Description
Site Name	The name of the Web site to start. For example: <code>DefaultWebSite</code> .

Optional properties:

Property Name	Description
Arguments	A newline-separated list of arguments to be added to the appcmd call. For example: /arg1:val1\n/arg2:val2\n/arg3:val3...
Command Path	The directory location of the AppCmd command-line executable. For example: C:\Windows\system32\inetsrv\.

Creating a Stop Site Step

This step stops a Web site in IIS.

Required properties:

Property Name	Description
Site Name	The name of the Web site to stop. For example: <code>DefaultWebSite</code> .

Optional properties:

Property Name	Description
Arguments	A newline-separated list of arguments to be added to the appcmd call. For example: /arg1:val1\n/arg2:val2\n/arg3:val3...
Command Path	The directory location of the AppCmd command-line executable. For example: C:\Windows\system32\inetsrv\.

Creating a Check if Site Exists Step

This step checks to see if a Web site exists in IIS.

Required properties:

Property Name	Description
Site Name	The name of the Web site to check. For example: <code>DefaultWebSite</code> .

Optional properties:

Property Name	Description
Arguments	A newline-separated list of arguments to be added to the appcmd call. For example: /arg1:val1\n/arg2:val2\n/arg3:val3...
Command Path	The directory location of the AppCmd command-line executable. For example: C:\Windows\system32\inetsrv\.

Creating a Check if Site is Stopped Step

This step checks to see if a Web site is stopped in IIS.

Required properties:

Property Name	Description
Site Name	The name of the Web site to check. For example: <code>DefaultWebSite</code> .

Optional properties:

Property Name	Description
Arguments	A newline-separated list of arguments to be added to the appcmd call. For example: /arg1:val1\n/arg2:val2\n/arg3:val3...
Command Path	The directory location of the AppCmd command-line executable. For example: C:\Windows\system32\inetsrv\.

Creating a Check if Site is Running Step

This step checks to see if a Web site is running in IIS.

Required properties:

Property Name	Description
Site Name	The name of the Web site to check. For example: <code>DefaultWebSite</code> .

Optional properties:

Property Name	Description
Arguments	A newline-separated list of arguments to be added to the appcmd call. For example: /arg1:val1\n/arg2:val2\n/arg3:val3...
Command Path	The directory location of the AppCmd command-line executable. For example: C:\Windows\system32\inetsrv\.

Creating a Create Virtual Directory Step

This step creates a virtual directory in IIS.

Required properties:

Property Name	Description
Parent Application	Application identifier under which this virtual directory should be created. This refers to the application pool name followed by the application name.
Virtual Path	The virtual path of the virtual directory.

Optional properties:

Property Name	Description
Path	The physical path of the virtual directory.
Arguments	A newline-separated list of arguments to be added to the appcmd call. For example: /arg1:val1\n/arg2:val2\n/arg3:val3...
Command Path	The directory location of the AppCmd command-line executable. For example: C:\Windows\system32\inetsrv\.

Creating a Delete Virtual Directory Step

This step deletes a virtual directory in IIS.

Required properties:

Property Name	Description
Virtual Directory Path	The virtual directory path or URL to be deleted. For example: Default Web Site/subdir.

Optional properties:

Property Name	Description
Arguments	A newline-separated list of arguments to be added to the appcmd call. For example: /arg1:val1\n/arg2:val2\n/arg3:val3...
Command Path	The directory location of the AppCmd command-line executable. For example: C:\Windows\system32\inetsrv\.

Creating a Create Application Pool Step

This step creates a new application pool in IIS.

Required properties:

Property Name	Description
Application Pool Name	The name of the application pool that you want to create.

Optional properties:

Property Name	Description
Arguments	A newline-separated list of arguments to be added to the appcmd call. For example: /arg1:val1\n/arg2:val2\n/arg3:val3...
Command Path	The directory location of the AppCmd command-line executable. For example: C:\Windows\system32\inetsrv\.

Creating a Delete Application Pool Step

This step deletes an application pool in IIS.

Required properties:

Property Name	Description
Application Pool Name	The name of the application pool that you want to delete.

Optional properties:

Property Name	Description
Arguments	A newline-separated list of arguments to be added to the appcmd call. For example: /arg1:val1\n/arg2:val2\n/arg3:val3...
Command Path	The directory location of the AppCmd command-line executable. For example: C:\Windows\system32\inetsrv\.

Creating a Start Application Pool Step

This step starts an application pool in IIS.

Required properties:

Property Name	Description
Application Pool Name	The name of the application pool that you want to start.

Optional properties:

Property Name	Description
Arguments	A newline-separated list of arguments to be added to the appcmd call. For example: /arg1:val1\n/arg2:val2\n/arg3:val3...
Command Path	The directory location of the AppCmd command-line executable. For example: C:\Windows\system32\inetsrv\.

Creating a Stop Application Pool Step

This step stops an application pool in IIS.

Required properties:

Property Name	Description
Application Pool Name	The name of the application pool that you want to stop.

Optional properties:

Property Name	Description
Arguments	A newline-separated list of arguments to be added to the appcmd call. For example: /arg1:val1\n/arg2:val2\n/arg3:val3...
Command Path	The directory location of the AppCmd command-line executable. For example: C:\Windows\system32\inetsrv\.

Creating a Recycle Application Pool Step

This step recycles an application pool in IIS.

Required properties:

Property Name	Description
Application Pool Name	The name of the application pool that you want to recycle.

Optional properties:

Property Name	Description
Arguments	A newline-separated list of arguments to be added to the appcmd call. For example: /arg1:val1\n/arg2:val2\n/arg3:val3...
Command Path	The directory location of the AppCmd command-line executable. For example: C:\Windows\system32\inetsrv\.

Creating an AppCmd Step

This step runs an arbitrary appcmd command.

Required properties:

Property Name	Description
Command	The command to run using appcmd.
Object Type	The object type that the command should run against.

Optional properties:

Property Name	Description
Identifier	The identifier for the object, if required.
Arguments	A newline-separated list of arguments to be added to the appcmd call. For example: /arg1:val1\n/arg2:val2\n/arg3:val3...
Command Path	The directory location of the AppCmd command-line executable. For example: C:\Windows\system32\inetsrv\.

Chapter 13: Microsoft IIS MS-Deploy Plug-in

Microsoft IIS is a Web server application packaged with Windows Server for hosting websites, services, and applications.

IIS MSDeploy, also known as Web Deploy, enables you to:

- Deploy Web applications and websites to IIS servers
- Detect differences between source and destination content
- Synchronize IIS servers

It makes it easier to:

- Move servers to newer versions of IIS
- Backup servers

The MSDeploy plug-in automates:

- Synchronizing and deleting servers, sites, applications, and packages
- Starting, stopping, and recycling application pools through the MSDeploy application

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Synchronize
- Msdeploy
- Delete
- Start Application
- Stop Application
- Recycle Application

For information about IIS, see <http://www.iis.net/home>.

Creating a Synchronize Step

This step synchronizes two IIS objects.

Required properties:

Property Name	Description
Source Provider Type	The provider type for the source argument of the synchronize. Select a value from the list below. <ul style="list-style-type: none">• Application• Archive Directory• ContentPath• Directory• File• Manifest• MetaKey• Package• Server• Server 6.0• Site
Provider Source	The provider path and setting for the source. Use the syntax: <code>path, setAting1, settign2...</code>

Property Name	Description
Destination Provider Type	<p>The provider type for the destination argument of the synchronize. Select a value from the list below.</p> <ul style="list-style-type: none"> • Auto • Application • Archive Directory • ContentPath • Directory • File • Manifest • MetaKey • Package • Server • Server 6.0 • Site

Optional properties:

Property Name	Description
Provider Destination	The Provider path and setting for the destination provided using the syntax <code>path, setting1, setting2...</code>
Options String	A newline-separated list of arguments to concatenate onto the synchronize command. For example: <code>-setParam:hello,value=hi\n-setParam:goodbye,value=seeyou.</code>
Command Path	The directory location of the msdeploy.exe command line executable. For example: <code>C:\Program Files\IIS\Microsoft Web Deploy V2\.</code>

Creating an MSDeploy Step

This step runs an msdeploy command.

Required properties:

Property Name	Description
Verb	The verb (operation) to be performed by msdeploy.

Optional properties:

Property Name	Description
Source Provider Type	The provider type for the source argument of the command. This is required if the provider source is supplied.
Provider Source	The provider path and setting for the source given in the syntax <code>path, setting1, setting2...</code>
Destination Provider Type	The provider type for the destination argument of the command.
Provider Destination	The provider path and setting for the destination provided using the syntax <code>path, setting1, setting2...</code>
Options String	A newline-separated list of arguments to concatenate onto the synchronize command. For example: <code>-setParameter:hello,value=hi\n-setParam:goodbye,value=seeyou.</code>
Command Path	The directory location of the msdeploy.exe command line executable. For example: <code>C:\Program Files\IIS\Microsoft Web Deploy V2\.</code>

Creating a Delete Step

This step deletes an IIS object.

Required properties:

Property Name	Description
Provider Type	The provider type for the destination argument of the delete. Select a value from the list below. <ul style="list-style-type: none">• Application• Archive Directory• ContentPath• Directory• File• Manifest• MetaKey• Package• Site
Provider Destination	The provider path and setting for the destination supplied using the syntax <code>path,setting1,setting2...</code>

Optional properties:

Property Name	Description
Options String	A newline-separated list of arguments to concatenate onto the delete command. For example: <code>-setParam:hello,value=hi\n-setParam:goodbye,value=seeyou.</code>
Command Path	The directory location of the msdeploy.exe command line executable. For example: <code>C:\Program Files\IIS\Microsoft Web Deploy V2\.</code>

Creating a Stop Application Step

This step stops an application in IIS.

Required properties:

Property Name	Description
Target Name	The name of the Web site or application. For example: <code>website/MyApp</code> .

Optional properties:

Property Name	Description
Command Path	The directory location of the <code>msdeploy.exe</code> command line executable. For example: <code>C:\Program Files\IIS\Microsoft Web Deploy V2\</code> .

Creating a Recycle Application Step

This step stops an application in IIS.

Required properties:

Property Name	Description
Target Name	The name of the Web site or application. For example: <code>website/MyApp</code> .

Optional properties:

Property Name	Description
Command Path	The directory location of the <code>msdeploy.exe</code> command line executable. For example: <code>C:\Program Files\IIS\Microsoft Web Deploy V2\</code> .

Chapter 14: Oracle WebLogic WLDeploy Plug-in

Oracle WebLogic Server includes the `wldeploy` Ant task to enable you to perform `weblogic.Deployer` functions using attributes specified in an Ant XML file. You can use `wldeploy` along with other WebLogic Server Ant tasks to create a single Ant build script.

The WLDeploy plug-in enables you to deploy, undeploy, and redeploy Java applications and start and stop WebLogic servers and clusters.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Run WLDeploy
- Start Targets
- Stop Targets
- Check Targets
- Check Application on targets
- Wait for Application on targets
- List Applications on targets

For information about WLDeploy, see http://docs.oracle.com/cd/E13222_01/wls/docs103//programming/wldeploy.html.

Part 3: Build Management Plug-ins

This section contains the following information:

- [Chapter 15: Apache Ant Plug-in \[page 131\]](#)
- [Chapter 16: Apache Maven Plug-in \[page 133\]](#)
- [Chapter 17: MS Build Plug-in \[page 135\]](#)
- [Chapter 18: UrbanCode AnthillPro Plug-in \[page 137\]](#)

Chapter 15: Apache Ant Plug-in

Apache Ant is a Java-based build tool. It is typically used to build Java applications; however, it can also be used to build C or C++ applications. With Ant, the build process and its dependencies are described using XML.

Use the Ant plug-in during deployment to automate the execution of Ant tasks defined in a `build.xml` file.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Ant

For information about Ant, see <http://ant.apache.org/>.

Creating an Ant Step

This step executes an Ant script.

Required properties:

Property Name	Description
Ant Script File	The name of the Ant script file.
ANT_HOME	The path to the Ant installation to execute the Ant script. By default, the step uses the agent's ANT_HOME environment variable. This refers to the path up to (but not including) the bin folder.
JAVA_HOME	The path to the Java installation to execute Ant. By default, the agent's JAVA_HOME environment variable is used.

Optional properties:

Property Name	Description
Targets	The names of the targets to run in the Ant script file. If left empty, the default target is run.
Properties	The properties passed to Ant. The properties are made available by name in the Ant script. This is supplied as a newline-separated list in the format <code>name=value</code> .

Property Name	Description
Ant Properties	Ant-specific arguments. For example, <code>-v</code> is used to indicate verbose output. This is supplied as a newline-separated list.
JVM Properties	JVM-specific arguments. For example <code>-Xmx=512m</code> is used for max memory. This is supplied as a newline-separated list.
Script Content	The content of the Ant script. The value provided here is written to the Ant Script File, and then executed.

Chapter 16: Apache Maven Plug-in

Apache Maven is a software project management tool that can manage a project's build, reporting, and documentation.

The Maven plug-in enables you to resolve artifacts from a Maven repository.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Maven Resolve

For information about Maven, see <http://maven.apache.org/>.

Chapter 17: MS Build Plug-in

Microsoft Build Engine (MSBuild) automates the process of compiling source code, packaging, testing, deploying, and creating documentation. Typically MSBuild is used in conjunction with Microsoft Visual Studio.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Run MSBuild

For information about MSBuild, see [http://msdn.microsoft.com/en-us/library/ms171452\(v=vs.90\).aspx](http://msdn.microsoft.com/en-us/library/ms171452(v=vs.90).aspx).

Chapter 18: UrbanCode AnthillPro Plug-in

UrbanCode AnthillPro is a continuous integration server that automates the process of building and testing code in software projects.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Run Workflow
- Download Artifacts
- Assign Status

For information about AnthillPro, see <http://www.urbancode.com/html/products/anthillpro/>.

Part 4: Configuration Management and Repository Plug-ins

This section contains the following information:

- [Chapter 19: CollabNet Teamforge Plug-in \[page 141\]](#)
- [Chapter 20: Servicenow Plug-in \[page 143\]](#)
- [Chapter 21: Microsoft Sharepoint Plug-in \[page 145\]](#)
- [Chapter 22: Serena PVCS VM Plug-in \[page 147\]](#)
- [Chapter 23: Subversion Plug-in \[page 149\]](#)

Chapter 19: CollabNet Teamforge Plug-in

CollabNet Teamforge is an integrated Web-based platform that enables teams to work collaboratively through the various stages of the application life cycle, including the following activities: plan, code, track, build and test, lab management, release, report, and collaborate.

The TeamForge plug-in enables Serena Release Automation to create and modify artifacts and create file releases.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Comment on Tracker Artifacts
- Create Tracker Artifact
- Create Release
- Upload Release Files
- Download Release Files
- Upload Document Files

For information about TeamForge, see <http://www.collab.net/products/teamforge>.

Chapter 20: ServiceNow Plug-in

ServiceNow provides cloud-based services that automate IT operations.

The ServiceNow Change Management plug-in enables Serena Release Automation to check approvals and set the status of change requests by interacting with the Configuration Management Database (CMDB).

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Check Approval Status
- Check Change Request Status
- Set Change Request Status
- Check Change Request's Child Tasks Statuses
- Update Tasks Statuses
- Insert Row Into CMDB Table
- Delete Row From CMDB Table
- Delete Multiple Rows From CMDB Table

For information about ServiceNow, see <http://www.servicenow.com/>.

Chapter 21: Microsoft Sharepoint Plug-in

SharePoint is a collection of websites that provides intranet content management and document management facilities.

The Sharepoint plug-in enables Serena Release Automation to deploy Windows SharePoint (WSP) and Content Migration Package (CMP) packages.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Export SharePoint Content
- Deploy SharePoint CMP File
- Deploy SharePoint WSP File
- Activate SharePoint Feature
- Deploy SharePoint WSP File to Sandbox

For information about Sharepoint, see <http://www.microsoft.com/en-gb/business/products/sharepoint-2013.aspx>.

Chapter 22: Serena PVCS VM Plug-in

Serena PVCS Version Manager (PVCS VM) is a source revision control tool.

The PVCS VM plug-in provides support for checking out from PVCS.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- PVCS Export

For information about Serena PVCS, see <http://www.serena.com/index.php/en/products/pvcs-vm/>.

Creating a PVCS Export Step

This step performs a PVCS checkout.

Required properties:

Property Name	Description
PCLI Path	The path to the PCLI executable.
Database Path	The path to the PVCS database.
Base Path	The base path of checked-out files.
Project Path	The path to the PVCS project (folder) relative to the database.

Optional properties:

Property Name	Description
Branch	The branch from which to pull the latest files.
Label	The label from which to pull the latest files.
Promotion Group	The promotion group from which to pull the latest files.
Value for Clean Workspace	If selected, all files are erased from the workspace before an export is performed.
User	The authentication user.
Password	The user's password.



Important: You must supply a **Branch**, **Label**, or **Promotion Group** property value, but only one these properties should be used.

Chapter 23: Subversion Plug-in

Subversion is an open source Version Control System (VCS).

The Checkout operation is used to pull files from a server onto a user's machine, allowing the user to enforce version control on the files obtained. The Export operation is similar except that it does not allow version control to be enforced on the repository obtained.

The Subversion plug-in enables Serena Release Automation to check out and export code from Subversion.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Svn Export

For information about Subversion, see <http://subversion.apache.org/>.

Part 5: Database Management Plug-ins

This section contains the following information:

- [Chapter 24: DBDeploy Plug-in \[page 153\]](#)
- [Chapter 25: DBUpgrader Plug-in \[page 155\]](#)
- [Chapter 26: SQLCMD Plug-in \[page 157\]](#)
- [Chapter 27: SQL*Plus Plug-in \[page 159\]](#)
- [Chapter 28: SQL-JDBC Plug-in \[page 161\]](#)

Chapter 24: DBDeploy Plug-in

The DBDeploy plug-in enables Serena Release Automation to upgrade a database. This plug-in consists of the following step for you to add to your processes in Serena Release Automation:

- Run DbDeploy

For information about DBDeploy, see <http://dbdeploy.com/>.

Creating a Run DbDeploy Step

You may use this step to upgrade your database.

To create a Run DbDeploy step:

1. Enter a `Name` for your Run DbDeploy step.
2. Specify values for the following required properties:

Property	Description
Driver Classname	Class name of the driver for the database.
DB Driver Jar Base Path	Location of the DB driver jar files.
DB Driver Jars	Name of the database driver .jar file. Supports the * wildcard.
URL	Database URL. For example, <code>jdbc:hsqldb:hsq://localhost/xd</code>
User	Username for the database login.
Password	Password for the username for the database login.
SQL File path	The path to the directory within which the SQL files reside.
Changelog table name	Name of the change log table to use. This is useful if the DDL and DML need to be separate when deploying to replicated environments.
SQL statement delimiter	The delimiter to use to separate scripts into statements. Default is a semicolon ";"

Property	Description
SQL statement delimiter type	To split on the delimiter whenever it occurs, specify <code>normal</code> . To split on the delimiter only if it is featured on a line by itself, specify <code>row</code> . Default is <code>normal</code> .
Line Ending	<p>How lines should be separated in SQL statements that are issued by way of JDBC. By default, the step uses the appropriate line ending for the platform it is running on and this is normally satisfactory. However, due to a bug in some Oracle drivers, the Windows default of <code>CRLF</code> may not always work.</p> <p>This property takes the values:</p> <ul style="list-style-type: none">• <code>platform</code>• <code>cr</code>• <code>lf</code>• <code>crlf</code>

3. (Optional) You may choose to specify the following:
 - a. The `Output File` property can be used to switch to output script mode. The name of the script that DBDeploy will output. This should include a full or relative path.
 - b. The `DB Type` is required if script mode is used. This refers to the target DBMS.
 - c. The `Template Directory` is the directory from which to read customized template scripts. This is only relevant in script mode.

Chapter 25: DBUpgrader Plug-in

The DBUpgrader plug-in helps manage database changes, including schema changes and rollbacks.

DBUpgrader uses a proprietary XML format in conjunction with a version table in your database. The XML file is used to associate changes with versions and the database table tracks the changes that are applied. The plug-in logic is used when performing upgrades for AnthillPro and Serena Release Automation applications.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Upgrade DB
- Rollback DB

Chapter 26: SQLCMD Plug-in

Microsoft SQL Server SQLCMD is a command line application that exposes the management features of SQL Server. It comes with Microsoft SQL Server.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Run SQLCmd scripts

For information about SQLCMD, see <http://msdn.microsoft.com/en-gb/library/bb545450.aspx>.

Chapter 27: SQL*Plus Plug-in

SQL*Plus provides access to the Oracle database using the command line to enter SQL, PL/SQL, SQL *Plus and operating system commands.

The SQL*Plus plug-in enables Serena Release Automation to execute SQL scripts during a deployment.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Run SQLPlus script

For information about SQL*Plus, see http://docs.oracle.com/cd/B19306_01/server.102/b14357/toc.htm.

Creating a Run SQLPlus Script Step

This step runs a SQLPlus script.

Required properties:

Property Name	Description
SQLPlus Executable	The full path to the sqlplus executable, the scripts to be run, or the command to be run, if it is on the path.
Username	The username used to run the scripts.
Password	The password of the user.
Connection ID	The connection ID to be used. For example: <code>localhost:1521/ORCL</code> .
SQL Files	A newline-separated list of SQL files to be executed. Order is preserved. Supports '*' and '?' as wildcards.

Optional properties:

Property Name	Description
Oracle Home	The ORACLE_HOME environment variable value.

Chapter 28: SQL-JDBC Plug-in

The SQL-JDBC plug-in is database-independent. It allows users to execute SQL scripts in a specific order using JDBC drivers.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Execute SQL Scripts

Part 6: Integration Plug-ins

This section contains the following information:

- [Chapter 29: Informatica Plug-in \[page 165\]](#)
- [Chapter 30: Microsoft BizTalk Plug-in \[page 167\]](#)

Chapter 29: Informatica Plug-in

Informatica Data Services enables you to directly access and merge data across remote systems in your organization.

The Informatica plug-in is executed during deployment to migrate Informatica configurations, run scripts against the Informatica server, and create and deploy groups.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Create Static Deployment Group
- Create Dynamic Deployment Group
- Deploy Deployment Group
- Roll Back Deployment Group
- Validate Deployment Group
- Apply Label
- Run PMREP Command
- Import Objects

For information about Informatica, see <http://www.informatica.com/us/>.

Chapter 30: Microsoft BizTalk Plug-in

Microsoft Biztalk Server is an enterprise service bus (ESB). It enables remote systems to communicate based on business rules defined in the BizTalk application.

The BizTalk plug-in is used during deployment to import BizTalk applications and bindings, uninstall BizTalk applications, remove BizTalk applications, and execute BizTalk runbook automations.

The BizTalk plug-in relies on powershell and the bts executable to import new or updated BizTalk applications, start and stop BizTalk applications, and remove or uninstall BizTalk Applications.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Export BizTalk Application
- Export BizTalk Application Bindings
- Import or Upgrade BizTalk Application
- Restart BizTalk host
- Remove BizTalk Application
- Uninstall BizTalk Application
- Start BizTalk Application

For information about BizTalk, see <http://www.microsoft.com/en-us/biztalk/default.aspx>.

Part 7: Network Management Plug-ins

This section contains the following information:

- [Chapter 31: Citrix NetScaler Plug-in \[page 171\]](#)
- [Chapter 32: F5 BIG-IP Plug-in \[page 173\]](#)
- [Chapter 33: Nagios XI Plug-in \[page 175\]](#)

Chapter 31: Citrix NetScaler Plug-in

Citrix NetScaler makes applications and cloud-based services run faster by offloading application and database servers, accelerating their performance, and integrating security. NetScaler can be deployed before deploying Web and database servers to combine high-speed load balancing and content switching, data compression, content caching, SSL acceleration, network optimization, application visibility, and application security.

The NetScaler plug-in allows uDeploy to enable and disable servers, server groups, and services on NetScaler servers.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Enable Servers
- Disable Servers
- Enable Services
- Disable Services
- Enable Service Groups
- Check Service Group Status
- Check Service Group Bindings
- Disable Service Groups

For information about Citrix Netscaler, see <http://www.citrix.com/products/netscaler-application-delivery-controller/overview.html>.

Chapter 32: F5 BIG-IP Plug-in

F5 BIG-IP provides functionality such as network load balancing, access control, and application security for Web-based applications.

The F5 BIG-IP plug-in enables you to execute various F5 BIG-IP steps as a part of the deployment process.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Disable node in pool
- Disable node in all pools
- Enable node in pool
- Enable node in all pools
- Add node to pool
- Remove node from pool
- Create pool
- Delete pool

For information about F5 BIG-IP, see <http://www.f5.com/>.

Chapter 33: Nagios XI Plug-in

Nagios XI enables you to monitor critical infrastructure components such as applications, services, operating systems, network protocols, system metrics and network infrastructure.

The Nagios XI plug-in automates the enabling and disabling of notifications for groups and hosts during deployment.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Disable Host Notifications
- Enable Host Notifications
- Disable Group Notifications
- Enable Group Notifications

For information about Nagios XI, see <http://www.nagios.com/products/nagiosxi>.

Part 8: Quality and Test Management Plug-ins

This section contains the following information:

- [Chapter 34: Atlassian Jira Plug-in \[page 179\]](#)
- [Chapter 35: Deploy Tools Plug-in \[page 181\]](#)
- [Chapter 36: HP Quality Center Plug-in \[page 183\]](#)
- [Chapter 37: QuickTest Pro Plug-in \[page 193\]](#)
- [Chapter 38: Rally Plug-in \[page 195\]](#)
- [Chapter 39: Selenium Plug-in \[page 197\]](#)

Chapter 34: Atlassian Jira Plug-in

Atlassian Jira is used primarily for bug tracking, issue tracking, and project management.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Add Comments
- Create Issue
- Update Issue
- Check Status
- Publish Issue Report

For information about Jira, see <http://www.atlassian.com/software/jira/overview>.

Chapter 35: Deploy Tools Plug-in

Deploy Tools enables you to verify the deployment of an application through an HTTP interface.

The Deploy Tools plug-in provides automated post-deployment verification.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Verify Deployment
- Wait for web page

Chapter 36: HP Quality Center Plug-in

HP Quality Center (HP QC) provides quality assurance through requirements management, release and cycle management, test management, defects management, and reporting.

The HP QC plug-in automates the creating and deleting of issues in HP QC among other tasks.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Create Issue
- Update Issues
- Add Comments
- Publish Issue Report
- Run Test Set
- Publish Test Set Report
- Check Status
- Query Defects

For information about HP QC, see <http://www8.hp.com/uk/en/software-solutions/software.html?compURI=1172141>.

Creating a Create Issue Step

This step creates a new defect in HP Quality Center.

Required properties:

Property Name	Description
Server URL	The base URL of the Quality Center instance. In usually ends with /qcb.in.
Username	The username to authenticate with Quality Center.
Domain	The domain for the defect in Quality Center
Project	The project for the defect in Quality Center.
Project Key	The project key in Quality Center.

Property Name	Description
Assignee	The user the new defect is assigned to.
Summary	A summary for the new defect. By default, the max length is 255.
Detected By	The person who found the defect.
Detected On Date	The date the defect was detected. This is supplied using the format <code>MM/DD/YYYY</code> .
Detected In Version	The version in which the defect was detected.
Reproducible	Whether the defect is reproducible or not (Y/N).
Subject	The subject of the defect.
Severity	The severity of the new defect. The default values are: 1-Low, 2-Medium, 3-High, 4-Very High, and 5-Urgent.

Optional properties:

Property Name	Description
Password	The password to authenticate with Quality Center.
Password Script	If a property or script is used to obtain the password, enter it here and leave the password property empty.
Priority	The priority of the new defect. The default values are: 1-Low, 2-Medium, 3-High, 4-Very High, and 5-Urgent.
Status	The status of the new defect. The default values are: Closed, Fixed, New, Open, Rejected and Reopen. The default value is <code>New</code> .
Additional Fields	Additional defect fields to add. This is provided as a newline-separated list in the form <code>name=value</code> . Uses Java's <code>java.util.Properties</code> format.

Creating an Update Issues Step

This step updates one or more defects in HP Quality Center.

Required properties:

Property Name	Description
Server URL	The base URL of the Quality Center instance. In usually ends with /qcbn.
Username	The username to authenticate with Quality Center.
Domain	The domain for the defect in Quality Center
Project	The project for the defect in Quality Center.
Defect IDs	Comma-separated list of defect IDs to update.
Fail Mode	The action that should be taken when a defect to be updated is not found in Quality Center. Select one of the following values: <ul style="list-style-type: none">• Fail-fast: Fail the step immediately if a defect is not found.• Fail: Fail the step after attempting to update all defects.• Warn: Log a warning when a defect is not found.

Optional properties:

Property Name	Description
Password	The password to authenticate with Quality Center.
Password Script	If a property or script is used to obtain the password, enter it here and leave the password property empty.
Assignee	The user the new defect is assigned to.
Summary	A summary for the new defect. By default, the max length is 255.
Priority	The priority of the new defect. The default values are: 1-Low, 2-Medium, 3-High, 4-Very High, and 5-Urgent.
Severity	The severity of the new defect. The default values are: 1-Low, 2-Medium, 3-High, 4-Very High, and 5-Urgent.
Status	The status of the new defect. The default values are: Closed, Fixed, New, Open, Rejected and Reopen. The default value is New .

Property Name	Description
Additional Fields	Additional defect fields to add. This is provided as a newline-separated list in the form <code>name=value</code> . Uses Java's <code>java.util.Properties</code> format.
Comment	A comment to add to the update.

Creating an Add Comments Step

This step adds comments to a defect in HP Quality Center.

Required properties:

Property Name	Description
Server URL	The base URL of the Quality Center instance. In usually ends with <code>/qcbin</code> .
Username	The username to authenticate with Quality Center.
Domain	The domain for the defect in Quality Center
Project	The project for the defect in Quality Center.
Defect IDs	Comma-separated list of defect IDs to update.
Fail Mode	The action that should be taken when a defect to be updated is not found in Quality Center. Select one of the following values: <ul style="list-style-type: none"> Fail-fast: Fail the step immediately if a defect is not found. Fail: Fail the step after attempting to update all defects. Warn: Log a warning when a defect is not found.

Optional properties:

Property Name	Description
Password	The password to authenticate with Quality Center.
Password Script	If a property or script is used to obtain the password, enter it here and leave the password property empty.

Property Name	Description
Additional Comments	Additional information you can add to the Quality Center comment along with the commit comment.

Creating a Publish Issue Report Step

This step creates a report of Quality Center defects from a list of defect IDs.

Required properties:

Property Name	Description
Server URL	The base URL of the Quality Center instance. In usually ends with <code>/qcbn.</code>
Username	The username to authenticate with Quality Center.
Domain	The domain for the defect in Quality Center
Project	The project for the defect in Quality Center.
Defect IDs	Comma-separated list of defect IDs to update.
Output File	The output file for results.
Fail Mode	The action that should be taken when a defect to be updated is not found in Quality Center. Select one of the following values: <ul style="list-style-type: none"> • Fail-fast: Fail the step immediately if a defect is not found. • Fail: Fail the step after attempting to update all defects. • Warn: Log a warning when a defect is not found.

Optional properties:

Property Name	Description
Password	The password to authenticate with Quality Center.
Password Script	If a property or script is used to obtain the password, enter it here and leave the password property empty.

Creating a Run Test Set Step

This step runs a test set using HP Quality Center.

Required properties:

Property Name	Description
Server URL	The base URL of the Quality Center instance. In usually ends with /qcbn.
Username	The username to authenticate with Quality Center.
Domain	The domain for the defect in Quality Center
Project	The project for the defect in Quality Center.
Folder	The folder in Quality Center in which the test set resides. Usually starts with <code>Root</code> .
Test Set	The test set to run.

Optional properties:

Property Name	Description
Password	The password to authenticate with Quality Center.
Password Script	If a property or script is used to obtain the password, enter it here and leave the password property empty.
Remote Host	The host that runs the tests. If this is left empty, then the tests will run locally.

Creating a Publish Test Set Report Step

This step publishes an HP Quality Center Test Set Report.

Required properties:

Property Name	Description
Server URL	The base URL of the Quality Center instance. In usually ends with /qcbn.
Username	The username to authenticate with Quality Center.

Property Name	Description
Domain	The domain for the defect in Quality Center
Project	The project for the defect in Quality Center.
Folder	The folder in Quality Center in which the test set resides. Usually starts with <code>root</code> .
Test Set	The test set results to publish.
Report Name	The name that will be given to the report.
Output File	The output file for results.

Optional properties:

Property Name	Description
Password	The password to authenticate with Quality Center.
Password Script	If a property or script is used to obtain the password, enter it here and leave the password property empty.

Creating a Check Status Step

This step ensures the status of issues are in the expected state.

Required properties:

Property Name	Description
Server URL	The base URL of the Quality Center instance. In usually ends with <code>/qcbn</code> .
Username	The username to authenticate with Quality Center.
Domain	The domain for the defect in Quality Center
Project	The project for the defect in Quality Center.
Defect IDs	A comma-separated list of defect IDs to update in HP Quality Center.

Property Name	Description
Expected Status	The status that the issues are expected to have.
Fail Mode	The action that should be taken when a defect to be updated is not found in Quality Center. Select one of the following values: <ul style="list-style-type: none"> • Fail-fast: Fail the step immediately if a defect is not found. • Fail: Fail the step after attempting to update all defects. • Warn: Log a warning when a defect is not found.

Optional properties:

Property Name	Description
Password	The password to authenticate with Quality Center.
Password Script	If a property or script is used to obtain the password, enter it here and leave the password property empty.

Creating a Query Defects Step

This step queries defects in HP Quality Center.

Required properties:

Property Name	Description
Server URL	The base URL of the Quality Center instance. In usually ends with /qcbn.
Username	The username to authenticate with Quality Center.
Domain	The domain for the defect in Quality Center
Project	The project for the defect in Quality Center.

Optional properties:

Property Name	Description
Password	The password to authenticate with Quality Center.
Password Script	If a property or script is used to obtain the password, enter it here and leave the password property empty.
Match Criteria	A newline-separated list of criteria names to filter by. For example: <code>Created in Application = xxxxxxxxxxxx.</code>
Return Fields	A newline-separated list of fields to return.

Chapter 37: QuickTest Pro Plug-in

QuickTest Pro enables you to automate functional and regression tests. It can perform operations such as mouse clicks or keyboard events on objects in a GUI or Web page.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Run QuickTestPro tests

For information about QuickTest Pro, see <http://www.th-hp.com/hp-products.php?gclid=CNzr7eaGm7cCFRLLtAodoT4APg#QTP>.

Chapter 38: Rally Plug-in

Rally is an agile project management tool that help large enterprises perfect the art of agile development and bring products to market faster.

The Rally plug-in enables you to update Rally issues for bug or feature tracking.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Add Comments
- Change Status
- Create Defect
- Change Rally Artifact Property

For information about Rally, see <http://www.rallydev.com/>.

Chapter 39: Selenium Plug-in

Selenium is a testing tool which automates Web applications. It can also be used to automate basic browser functions and administration tasks.

The Selenium plug-in runs Selenium Remote Control (RC) commands using a Selenium HTML test suite file.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Run Test Suite

For information about Selenium, see <http://www.seleniumsoftware.com/>.

Part 9: Release Automation Management Plug-ins

This section contains the following information:

- [Chapter 40: BMC ARA Plug-in \[page 201\]](#)
- [Chapter 41: Serena RA Application Plug-in \[page 203\]](#)
- [Chapter 42: Serena RA Component Plug-in \[page 205\]](#)
- [Chapter 43: Serena RA Configuration Management Plug-in \[page 207\]](#)
- [Chapter 44: Serena RA Environment Plug-in \[page 209\]](#)
- [Chapter 45: Serena RA Resource Plug-in \[page 211\]](#)
- [Chapter 46: Serena Release Automation System Plug-in \[page 213\]](#)
- [Chapter 47: Serena RA Version Plug-in \[page 215\]](#)
- [Chapter 48: Serena RA VFS Plug-in \[page 217\]](#)

Chapter 40: BMC ARA Plug-in

BMC Application Release Automation (ARA) enables consistent, auditable deployment processes for multi-tier applications and their supporting infrastructure across physical and virtual platforms.

The BMC ARA plug-in enables you to update a server, take a snapshot of a server, and issue a deliver.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Snapshot Server
- Preview
- Install
- Generic Rundeliver

For information about BMC ARA, see <http://www.bmc.com/products/product-listing/bmc-bladelogic-application-release-automation.html>.

Chapter 41: Serena RA Application Plug-in

Serena Release Automation (RA) applications initiate component deployments; they bring together components with their deployment targets, and orchestrate multi-component deployments.

The Serena RA Application plug-in enables you to create and manage Serena RA applications.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Create Application
- Create Application Process
- Add Component To Application
- Create Application Property
- Create Application Role
- Run Application Process
- Check If Application Exists
- Add User To Role
- Add Group To Role

For information about Serena Release Automation applications, see the *Serena Release Automation User's Guide*.

Chapter 42: Serena RA Component Plug-in

Serena Release Automation (RA) components represent deployable items along with user-defined processes that operate on them, usually by deploying them.

The Serena RA Component plug-in enables you to create and manage Serena RA components.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Create Component
- Create Component Property
- Create Component Role
- Add User To Role For Component
- Add Group To Role For Component
- Check If Component Exists

For information about Serena Release Automation components, see the *Serena Release Automation User's Guide*.

Chapter 43: Serena RA Configuration Management Plug-in

Serena Release Automation configuration templates typically contain configuration data for server configurations, but the data can be for any purpose.

The Serena RA Configuration Management plug-in enables you to download and upload configuration templates to and from Serena Release Automation.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Install Template

For information about Serena Release Automation configuration templates, see the *Serena Release Automation User's Guide*.

Chapter 44: Serena RA Environment Plug-in

Serena RA environments are user-defined collections of resources that host applications. The Serena RA Environment plug-in enables you to create and manage Serena RA environments.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Create Environment
- Create Component Mapping
- Create Environment Property
- Get Environment Properties
- Get Component Environment Properties
- Inactivate Environment
- Set Component Environment Property
- Add User To Role For Environment
- Add Group To Role For Environment
- Verify Inventory Status
- Check If Environment Exists

For information about Serena Release Automation environments, see the *Serena Release Automation User's Guide*.

Chapter 45: Serena RA Resource Plug-in

Serena RA resources represent deployment targets, such as a physical machines, virtual machines, databases, and J2EE containers.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Create Resource
- Add Resource to Group
- Create Resource Group
- Create Dynamic Resource Group
- Add Name Condition To Group
- Add Property Condition To Group
- Set Agent Property
- Set Resource Property
- Set Resource Role Property
- Check If Resource Has Role
- Add Role To Resource
- Remove Role From Resource
- Delete Resource Group
- Delete Resource
- Delete Agent
- Delete Many Resources
- Delete Many Agents
- Get Agent Property
- Get Resource Property
- Get Resource Role Property
- Delete Resource Inventory For Component
- Check If Resource Exists
- Add User To Role For Resource

- Add Group To Role For Resource
- Get Component Version For Resource
- Wait For Resources
- Add User To Role For Resource Group
- Add Group To Role For Resource Group

For information about Serena Release Automation resources, see the *Serena Release Automation User's Guide*. .

Chapter 46: Serena Release Automation System Plug-in

The Serena Release Automation System plug-in enables you to enter system, or global, properties to be used by a process. This plug-in consists of the following step for you to add to your processes in Serena Release Automation:

- Create System Property

For information about Serena Release Automation System Properties, see the *Serena Release Automation User's Guide*.

Creating a Serena Release Automation System Property Step

You may use this step to enter system, or global, properties to be used by a process..

To create a Serena Release Automation System Property step:

1. Enter a `Name` for your Serena Release Automation System Property step.
2. In the `Property Name` field, enter the name of the Serena Release Automation system property to be set by this step.
3. (Optional) You may choose to specify the following:
 - a. The `Property Value` to which you want the Serena Release Automation system property set by this step.
 - b. `Secure` the Serena Release Automation system property so that it is stored encrypted and its display is obscured in the Serena Release Automation user interface.

Chapter 47: Serena RA Version Plug-in

Serena Release Automation provides component version control using the Versioned File Storage (VFS) repository. CodeStation is the name of the service that manages and stores artifacts in the repository.

The Serena RA Version plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Add Status to Version
- Remove Status From Version
- Create Version

For information about Serena Release Automation component version control and the CodeStation repository, see the *Serena Release Automation User's Guide*.

Chapter 48: Serena RA VFS Plug-in

Serena Release Automation Versioned File Storage (VFS) enables you to add artifacts to Serena Release Automation version repository, CodeStation.

The Serena RA VFS plug-in enables you to upload to, download to, and verify artifacts in a Serena Release Automation VFS artifact repository.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Upload Artifacts
- Download Artifacts
- Verify Local Artifacts

For information about Serena Release Automation VFS, see the *Serena Release Automation User's Guide*.

Part 10: System Tools and Scripting Plug-ins

This section contains the following information:

- [Chapter 49: 7zip Plug-in \[page 221\]](#)
- [Chapter 50: CA AutoSys Plug-in \[page 223\]](#)
- [Chapter 51: FileUtils Plug-in \[page 225\]](#)
- [Chapter 52: Groovy Plug-in \[page 227\]](#)
- [Chapter 53: Microsoft Message Queuing \(MSMQ\) Plug-in \[page 229\]](#)
- [Chapter 54: Microsoft Software Installer \(MSI\) Plug-in \[page 231\]](#)
- [Chapter 55: Microsoft Windows Service Control Manager Plug-in \[page 233\]](#)
- [Chapter 56: Windows System Tools Plug-in \[page 235\]](#)
- [Chapter 57: Red Hat Package Manager \(RPM\) Plug-in \[page 237\]](#)
- [Chapter 58: Shell Plug-in \[page 239\]](#)
- [Chapter 59: System Information Plug-in \[page 241\]](#)

Chapter 49: 7zip Plug-in

7zip is an open source file archiver that combines a number of files together into a single file for easier transportation and storage. The 7zip plug-in provides the following step for you to add to your processes in Serena Release Automation:


- Extract Archive


For information about 7zip, see <http://www.7-zip.org/>.

Creating a 7zip Extract Archive Step

You may use this step to extract files from a 7zip archive.

To create a 7zip extract archive step:

1. Enter the `Name` for your extract archive step.
2. Use the `Include Files` field to list the file filters you want this step to use to select the files to include.
 -  **Important:** Files must each be listed on a new line separated from the next.
3. Specify from where the files should be extracted in the `Extract Directory` field.
4. (Optional) You may choose to specify the following:
 - a. The `Directory Offset` relative to the current working directory where the step should run.
 - b. Use the `Exclude Files` field to list file filters you want this step to exclude from using.

-  **Important:** Files must each be listed on a new line separated from the next.

Chapter 50: CA AutoSys Plug-in

CA Workload Automation AutoSys Edition is a workload automation tool for defining, scheduling, and monitoring jobs.

The AutoSys plug-in provides steps for integrating with a CA Workload Automation AE server.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Send Autosys Event
- Check Autosys Server
- Execute JIL Script

For information about AutoSys, see <http://www.ca.com/us/products/detail/CA-Workload-Automation-AE.aspx>.

Chapter 51: FileUtils Plug-in

FileUtils is a file utility plug-in that automates folder and file level tasks.

The FileUtils plug-in includes steps for creating and deleting directories and replacing tokens in a file.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Replace Tokens
- Untar Tarball
- Unzip
- Update Property File
- Copy Directory
- Move Directory
- Synchronize Directories
- Create Directories
- Check if Directory exists
- Delete Files and Directories
- Create File
- Update INI File
- Monitor File Contents
- Flip Line Endings
- Create Zip Archive
- Update XML File with XPath
- Read Property File

Chapter 52: Groovy Plug-in

Groovy is an object-oriented programming language for the Java platform. It is a dynamic language that can be used as a scripting language.

The Groovy plug-in provides steps for executing user-defined Groovy scripts.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Run Groovy Script

For information about Groovy, see <http://groovy.codehaus.org/>.

Chapter 53: Microsoft Message Queuing (MSMQ) Plug-in

Microsoft Message Queuing (MSMQ) technology enables applications running at different times to communicate across heterogeneous networks and systems that may be temporarily offline. Applications send and read messages to and from queues. A queue can hold messages that are generated by multiple sending applications and read by multiple receiving applications.

The MSMQ plug-in enables you to create steps for starting, stopping, and creating message queues in Microsoft MQ.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Create Message Queue
- Delete Message Queue
- Grant All Queue Permissions
- Grant Queue Permissions
- Revoke All Queue Permissions
- Revoke Queue Permissions

For information about MSMQ, see [http://msdn.microsoft.com/en-us/library/windows/desktop/ms711472\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/ms711472(v=vs.85).aspx).

Chapter 54: Microsoft Software Installer (MSI) Plug-in

Microsoft Software Installer (MSI) is a software component used for the installation, maintenance, and removal of software on Microsoft Windows systems.

The MSI plug-in installs and uninstalls MSI and runs MSIexec.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Install MSI
- Uninstall MSI
- Execute msiexec

For information about MSI, see <http://support.microsoft.com/kb/310598>.

Chapter 55: Microsoft Windows Service Control Manager Plug-in

Windows Service Control Manager stores information about the installed services and their status. It can start and stop services, transmit control requests to running services, and lock and unlock the service database.

The Service Control Manager plug-in is used during the deployment to automate Windows services.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Create Service
- Delete Service
- Start Service
- Stop Service
- Check If Service Exists
- Check If Service Stopped
- Check If Service Running
- Enable Service
- Disable Service

For information about Windows Service Control Manager, see [http://msdn.microsoft.com/en-us/library/d56de412\(v=vs.80\).aspx](http://msdn.microsoft.com/en-us/library/d56de412(v=vs.80).aspx).

Chapter 56: Windows System Tools Plug-in

The Windows System Tools plugin automates various Windows system tasks.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Modify ACLs
- Log off Notification
- Reboot
- Check File Contents
- Add or Overwrite Registry Key
- Export Registry Subkey to File
- Import Registry Files
- Configure DCOM Settings
- Create Message Queues
- Enable Windows Features
- Disable Windows Features

Chapter 57: Red Hat Package Manager (RPM) Plug-in

The Red Hat Package Manager (RPM) is an open packaging system that runs on Red Hat Enterprise Linux and other Linux and UNIX systems. This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Install RPM
- Uninstall RPM
- Update RPM

For information about Red Hat Package Manager (RPM), see <http://www.rpm.org/>.

Creating an Install RPM Step

You may use this step to install RPM packages.

To create an install RPM step:

1. Enter the `Name` for your install RPM step.
2. Use the `RPM Packages` field to list the RPM packages you want this step to install.



Important: Packages must each be listed on a new line separated from the next.

3. (Optional) You may choose to specify the following: `Install Options` (also in a new-line separated list) to be used during the install.

Creating an Uninstall RPM Step

You may use this step to uninstall RPM packages.

To create an uninstall RPM step:

1. Enter the `Name` for your uninstall RPM step.
2. Use the `RPM Packages` field to list the RPM packages you want this step to uninstall.




Important: Packages must each be listed on a new line separated from the next.

3. (Optional) You may choose to specify the following: `Erase Options` (also as a new-line separated list) to be used during the uninstall.

Creating an Update RPM Step

You may use this step to update RPM packages.

To create an update RPM step:

1. Enter the `Name` for your update RPM step.
2. Use the `RPM Packages` field to list the RPM packages you want this step to install.
 **Important:** Packages must each be listed on a new line separated from the next.
3. (Optional) You may choose to specify the following: `Update Options` (also in a new-line separated list) to be used during the install.

Chapter 58: Shell Plug-in

A shell is a command language interpreter that is usually included with operating systems. A shell script is a series of commands.

The Shell plug-in enables you to run custom shell scripts during the deployment process. The most common use case for this plug-in is opening and running a shell script on the target machine. If the step is used within a larger process, ensure that you set the order correctly. For example, if you have to run a shell script prior to executing another process, you will need to add the Shell step above the other step.



Note: This plug-in is included in the standard plug-ins provided with Serena Release Automation and need not be loaded unless you need to upgrade or extend it.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Shell
- Shell (with xargs)

Chapter 59: System Information Plug-in

The System Information plug-in includes a variety of checks to perform against the operating system. You can use this plug-in to verify that a deployment can succeed or has succeeded.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Check Connectivity
- Check Environment Variable
- Check Available Disk Space
- Check Regex

Part 11: Virtual and Cloud Environment Management Plug-ins

This section contains the following information:

- [Chapter 60: Amazon EC2 Plug-in \[page 245\]](#)
- [Chapter 61: Microsoft Windows Azure Plug-in \[page 251\]](#)
- [Chapter 62: VMware vSphere ESXi Plug-in \[page 253\]](#)
- [Chapter 63: VMware vCenter Plug-in \[page 255\]](#)
- [Chapter 64: VMware Workstation Plug-in \[page 257\]](#)

Chapter 60: Amazon EC2 Plug-in

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides environment support in the cloud. This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Launch Instance
- Create Security Group
- Start Instances
- Stop Instances
- Terminate Instances
- Wait for Instances
- Associate IPs
- Register Instances with LoadBalance
- Deregister Instances with LoadBalance
- Get Public DNS

For information about Amazon EC2, see <http://aws.amazon.com>.

Creating an Amazon EC2 Launch Instance Step

The basic building blocks of Amazon EC2 are the *Amazon Machine Images* (AMI). An AMI is a template that contains a software configuration, such as an operating system, application server, or applications that you can run in an Amazon environment. Amazon has a variety of AMIs available and you can also create you own.

To launch an instance from AMI, you need to create an Amazon EC2 `Launch Instance` step.

To create an Amazon EC2 launch instance step:

1. Enter a `Name` for your Launch Instance step.
2. Specify values for the following required properties:

Property	Description
Access Key Id	The EC2 access key ID to use to log in.
Secret Key	The EC2 secret key.

Property	Description
# of instances	The number of EC2 instances to startup.
Instance Type	The type of instances to run. Allowed values are m1.small, m1.large, m1.xlarge, m2.xlarge, m2.4xlarge, c1.medium, c1.xlarge.
AMI ID	The AMI ID of the instances to be started.
AWS Jar	The full path to the AWS SDK .jar file.

3. You may choose to specify values for the following optional properties:

Property	Description
Security Group	A comma-separated list of security group names to use.
Availability Zone	The zone to start these instances in.
Keypair	The keypair to start these instances with.
User data	The user data to be passed to the instance.

Creating an Amazon EC2 Security Group Step

The Amazon EC2 security group acts as a firewall that controls the traffic allowed into a group of instances.

To create a security group step:

1. Enter a `Name` for your Security Group step.
2. Specify values for the following required properties:

Property	Description
Access Key Id	The EC2 access key ID to use to log in.
Secret Key	The EC2 secret key.
AWS Jar	The full path to the AWS SDK .jar file.

3. You may choose to specify values for the following optional properties:

Property	Description
Name	The name of the security group to create, if not using a file to add permissions.
Description	The description of the security group to create, if not using a file to add permissions.
VPC ID	The ID of the Virtual Private Cloud (VPC) to use, if not using a file to add permissions.
Definition File	<p>The file describing the security group allowed ips/ports. If left blank, an empty security group will be created. Otherwise, the file has the following format:</p> <pre><SecurityGroup name="name" description="description" vpcId="id_optional"> <ipPermission protocol="tcp" fromPort="nn" toPort="nn"> <ipRange value="0.0.0.0/0"/> </ipPermission> </SecurityGroup></pre>

Amazon EC2 Start, Stop, and Terminate Instance Steps

To start, stop, or terminate an Amazon EC2 instance, you pass the same set of parameters. These give the information needed to identify the instance that you want to start, stop, or terminate. The only difference is start and stop steps accept only one Instance ID whereas the terminate step accepts more than one instance ID.

To create a start, stop, or terminate instance step:

1. Enter a `Name` for your start, stop, or terminate instance step.
2. Specify values for the following required properties:

Property	Description
Access Key Id	The EC2 access key ID to use to log in.
Secret Key	The EC2 secret key.
instance ID	The Instance ID to start or stop, or for the terminate step, a list of instance IDs separated by commas.
AWS Jar	The full path to the AWS SDK .jar file.

Creating an Amazon EC2 Wait for Instance Step

Use to create a step that waits until the instance(s) specified in the `Instance IDs` field switch to the state specified in the `State` field.

To create a Wait for Instance step:

1. Enter a `Name` for your Wait for Instance step.
2. Specify values for the following required properties:

Property	Description
Access Key Id	The EC2 access key ID to use to log in.
Secret Key	The EC2 secret key.
Instance IDS	A comma-separated list of instance IDs that correspond to the IPs to associate.
State	The state for instances to be in.
Timeout	The timeout for this step, in milliseconds.
AWS Jar	The full path to the AWS SDK .jar file.

Creating an Amazon EC2 Associate IPs Step

In an AWS, an Elastic IP Address (EIP) enables you to reserve an IP address that you can then assign to any AMI instance you have running. If needed, at any time you can also change the assignment to a different instance.

According to Amazon, this feature is designed for "dynamic cloud computing". Once an EIP has been associated with an instance, it remains associated with that instance until you release it.

To create an associate IPs step:

1. Enter a `Name` for your Associate IPs step.
2. Specify values for the following required properties:

Property	Description
Access Key Id	The EC2 access key ID to use to log in.
Secret Key	The EC2 secret key.
IPS	A new-line separated list of elastic ips to associate with instances.

Property	Description
Instance IDS	A comma-separated list of instance IDs that correspond to the IPs to associate.
AWS Jar	The full path to the AWS SDK .jar file.

Creating an Amazon EC2 Register Instances with LoadBalance Step

Elastic Load Balancing automatically distributes incoming application traffic across multiple Amazon EC2 instances. It enables you to achieve even greater fault tolerance in your applications, seamlessly providing the amount of load balancing capacity needed in response to incoming application traffic. Elastic Load Balancing detects unhealthy instances within a pool and automatically reroutes traffic to healthy instances until the unhealthy instances have been restored.

You can enable Elastic Load Balancing within a single *Availability Zone* or across multiple zones for even more consistent application performance. Elastic Load Balancing can also be used in an Amazon Virtual Private Cloud (VPC) to distribute traffic between application tiers. For more details, see the *Amazon Elastic Compute Cloud* documentation.

To create an Amazon EC2 Register Instance with LoadBalance Step:

1. Enter a `Name` for your Register Instance with LoadBalance step.
2. Specify values for the following required properties:

Property	Description
Access Key Id	The EC2 access key ID to use to log in.
Secret Key	The EC2 secret key.
Load Balancer Name	The name of the load balancer in this EC2 account to register the instances from.
Instance Ids	Instance ID to start.
AWS Jar	The full path to the AWS SDK .jar file.

Creating an Amazon EC2 Deregister Instances with LoadBalance Step

The Deregister Instances With LoadBalancer works the same as the re-registration except that you have to use another step for it.

To create an Amazon EC2 Deregister Instance with LoadBalance Step:

1. Enter a `Name` for your Register Instance with LoadBalance step.

2. Specify values for the following required properties:

Property	Description
Access Key Id	The EC2 access key ID to use to log in.
Secret Key	The EC2 secret key.
Load Balancer Name	The name of the load balancer in this EC2 account to de-register the instances from.
Instance Ids	Instance ID to start.
AWS Jar	The full path to the AWS SDK .jar file.

Creating an Amazon EC2 Get Public DNS Step

You may get the list of public DNSs that a list of Amazon EC2 instances are running on.

To create an Amazon EC2 Get Public DNS Step:

1. Enter a `Name` for your Get Public DNS step.
2. Specify values for the following required properties:

Property	Description
Access Key Id	The EC2 access key ID to use to log in.
Secret Key	The EC2 secret key.
Instance Ids	A comma-separated list of instance IDs to be retrieved. The DNS list returned will be in the same order as the IDs given.
AWS Jar	The full path to the AWS SDK .jar file.

Chapter 61: Microsoft Windows Azure Plug-in

Microsoft Windows Azure is an open cloud platform that enables you to build, deploy, and manage applications across a global network of Microsoft-managed datacenters.

The Azure plug-in enables the starting, restarting, stopping, and deletion of Microsoft Azure cloud instances.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Delete VM
- Restart VM
- Start VM
- Stop VM

For information about Azure, see <http://www.windowsazure.com>.

Chapter 62: VMware vSphere ESXi Plug-in

VMware vSphere installs directly on top of a physical server and partitions it into multiple virtual machines that can run simultaneously, sharing the physical resources of the underlying server.

The VMware ESXi plug-in provides steps for setting up, starting, stopping, suspending, taking, and reverting to snapshots for VMware vSphere ESX and ESXi virtual machines.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Delete Snapshot
- Start VM
- Stop VM
- Suspend VM
- Take Snapshot
- Revert VM to Snapshot
- VM settings

For information about VMware vSphere ESX and ESXi, see <http://www.vmware.com/products/vsphere/esxi-and-esx/overview.html>.

Chapter 63: VMware vCenter Plug-in

VMware vCenter Server provides centralized management of VMware virtual machines from a single console.

The VMware vCenter plug-in provides steps for setting up, starting, stopping, suspending, cloning, taking, and reverting to snapshots for virtual machines defined in VMware vCenter.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Clone VM
- Delete Snapshot
- Start VM
- Stop VM
- Suspend VM
- Take Snapshot
- Revert VM to Snapshot
- VM settings

For information about VMware vCenter, see <http://www.vmware.com/products/vcenter-server/overview.html>.

Chapter 64: VMware Workstation Plug-in

VMware Workstation enables you to access virtual machines running on VMware vSphere, ESXi, or another copy of VMware Workstation.

The VMware Workstation plug-in provides steps for setting up, starting, stopping, suspending, taking, and reverting to snapshots for VMware Workstation virtual machines.

This plug-in consists of the following available steps for you to add to your processes in Serena Release Automation:

- Clone VM
- Delete Snapshot
- Start VM
- Stop VM
- Suspend VM
- Take Snapshot
- Revert VM to Snapshot

For information about VMware Workstation, see <http://www.vmware.com/products/workstation/overview.html>.

Part 12: Creating Your Own Plug-ins

This section contains the following information:

- [Chapter 65: Plug-in Creation \[page 261\]](#)

Chapter 65: Plug-in Creation

You may create your own plug-ins if there is not an existing plug-in that meets your needs.

- [Plug-in Creation Overview \[page 261\]](#)
- [The plugin.xml File \[page 263\]](#)
- [Plug-in Steps: <step-type> Element \[page 264\]](#)
- [<command> Element \[page 267\]](#)
- [The <post-processing> Element \[page 268\]](#)
- [Upgrading Plug-ins \[page 269\]](#)
- [The info.xml File \[page 270\]](#)

Plug-in Creation Overview

A plug-in consists of two mandatory XML files—`plugin.xml` and `upgrade.xml`—along with any supporting script files required by the plug-in.

The `plugin.xml` file defines the steps comprising the plug-in; a plug-in's functionality is defined by its steps. Each step is an independently configurable entity in the Serena Release Automation editor.

The `upgrade.xml` file is used to upgrade the plug-in to a new version. Optionally, you can include an `info.xml` file which contains a version ID and other information. Although optional, Serena recommends the use of the `info.xml` file.

A plug-in step is defined by a `<step-type>` element that contains: one `<properties>` element, one `<command>` element, and one `<post-processing>` element. The `<properties>` element is a container for `<property>` child elements, and can contain any number of `<property>` elements. Property values can be supplied at design- or run-time. The `<post-processing>` element provides error-handling capabilities and sets property values that can be used by other steps. The `<command>` element performs the step's function. The function can be defined completely by the element, or be constructed in part or entirely from the step's properties at design- or run-time.

In addition to a step's own properties, a command has access to properties set earlier by other steps within the process, to properties set by the application that invoked the component process, as well as to those on the target environment and resource. Step property values become unavailable once the component process ends.

Plug-in steps are performed by an agent installed in the target environment, which means that plug-ins can be written in any scripting language as long as the agent can access the required scripting tools on the host. Once a plug-in is created, upload it into Serena Release Automation to make it available to users.

To upload a plug-in

1. Create a ZIP archive that contains the XML files (plugin.xml and upgrade.xml) along with any scripts required by the plug-in.
2. Import the ZIP file with the Automation Plug-ins pane Administration > Automation > Automation Plugins > Load Plugin.

The plugin.xml File

A plug-in is defined with the plugin.xml file. The structure of this file consists of a `header` element and one or more `step-type` elements. The `header` identifies the plug-in. Each `step-type` element defines a step; steps are available to users in the Serena Release Automation process editor and used to construct component processes.

After the document type declaration, the `plugin` root element identifies the XML schema type, `PluginXMLSchema_v1.xsd`, which is used by all plug-ins. The following presents the basic structure of plugin.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin xmlns="http://www.serena.com/PluginXMLSchema_v1"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <header>
    <identifier id="plugin_id" version="version_number" name="Plug-in Name"/>
    <description/>
    <tag>Plugin_type/Plugin_subtype/Plugin_name</tag>
  </header>
  <step-type name="Step_Name">
    <description/>
    <properties>
      <property name="property_name" required="true">
        <property-ui type="textBox" label="Driver Jar"
          description="The full path to the jdbc driver jar to use."
          default-value="{p:resource/sqlJdbc/jdbcJar}"/>
      </property>
    </properties>
    <post-processing>
      <![CDATA[
        if (properties.get("exitCode") != 0) {
          properties.put("Status", "Failure");
        }
        else {
          properties.put("Status", "Success");
        }
      ]]>
    </post-processing>
    <command program="{path_to_tool}"
      <arg value="parameters_passed_to_tool"/>
      <arg path="{p:jdbcJar}"/>
      <arg file="command_to_run"/>
      <arg file="{PLUGIN_INPUT_PROPS}"/>
      <arg file="{PLUGIN_OUTPUT_PROPS}"/>
    </command>
  </step-type>
</plugin>
```

<header> Element

The mandatory `header` element identifies the plug-in and contains three child elements:

<header> Child Elements	Description
<identifier>	<p>This element's three attributes identify the plug-in:</p> <ul style="list-style-type: none"> • <i>version</i> API version (the version number used for upgrading plug-ins is defined in the info.xml file). • <i>id</i> Identifies the plug-in. • <i>name</i> The plug-in name appears on Serena Release Automation's web application Automation Plugins pane. <p>All values must be enclosed within single-quotes.</p>
<description>	Describes the plug-in; appears on Serena Release Automation's web application Automation Plugins pane.
<tag>	<p>Defines where the plug-in is listed within the Serena Release Automation editor's hierarchy of available plug-ins. The location is defined by a string separated by slashes. For example, the Tomcat definition is: <code>Application Server/Java/Tomcat</code>. The Tomcat steps will be listed beneath the Tomcat item, which in turn is nested within the other two.</p>

The following is a sample header definition:

```
<header>
  <identifier version="3" id="com.&company;.air.plugin.Tomcat"name="Tomcat"/>
  <description>
    The Tomcat plugin is used during deployments to execute Tomcat run-book
    automations and deploy or undeploy Tomcat applications.
  </description>
  <tag>Application Server/Java/Tomcat</tag>
</header>
```

Plug-in Steps: <step-type> Element

Plug-in steps are defined with the `step-type` element; each `step-type` represents a single step in the Serena Release Automation process editor. A `step-type` element has a name attribute and several child elements: `description`, `properties`, `command`, and `post-processing`.

The mandatory name attribute identifies the step. The description and name appear in Serana Release Automation's web application.

```
<step-type name="Start">
  <description>Start Apache HTTP server</description>
```

Step Properties: `<properties>` Element

The `properties` element is a container for properties which are defined with the `property` tag. Each step has a single `properties` element; a `properties` element can contain any number of `property` child elements.

A `property` tag has a mandatory `name` attribute, optional `required` attribute, and two child elements, `property-ui` and `value`, which are defined in the following table.

<property> Element table

<property> Child Elements	Description
<property- ui>	<p>Defines how the property is presented to users in the Serena Release Automation editor. This element has several attributes:</p> <ul style="list-style-type: none"> • <code>label</code> Identifies the property in the editor's Edit Properties dialog box. • <code>description</code> Text displayed to users in the associated roll-over help box. • <code>default-value</code> Property value displayed when the Edit Properties dialog box is displayed; used if unchanged. • <code>type</code> Identifies the type of widget displayed to users. Possible values are: <ul style="list-style-type: none"> ▪ <code>textBox</code> Enables users to enter an arbitrary amount of text, limited to 4064 characters. ▪ <code>textAreaBox</code> Enables users to enter an arbitrary amount of text (larger input area than <code>textBox</code>), limited to limited to 4064 characters. ▪ <code>secureBox</code> Used for passwords. Similar to <code>textBox</code> except values are redacted. ▪ <code>checkBox</code> Displays a check box. If checked, a value of <code>true</code> will be used; otherwise the property is not set. ▪ <code>selectBox</code> Requires a list of one or more values which will be displayed in a drop-down list box. Configuring a value is described below.
<value>	<p>Used to specify values for a <code>selectBox</code>. Each value has a mandatory <code>label</code> attribute which is displayed to users, and a value used by the property when selected. Values are displayed in the order they are defined.</p>

Here is a sample `<property>` definition:

```

<property name="onerror" required="true">
  <property-ui type="selectBox"
    default-value="abort"
    description="Action to perform when statement fails: continue, stop, abort."
    label="Error Handling"/>
  <value label="Abort">abort</value>
  <value label="Continue">continue</value>
  <value label="Stop">stop</value>
</property>

```

<command> Element

Steps are executed by invoking the scripting tool or interpreter specified by the `<command>` element. The `<command>` element's `program` attribute defines the location of the tool that will perform the command. It bears repeating that the tool must be located on the host and the agent invoking the tool must have access to it. In the following example, the location of the tool that will perform the command—the Java-based scripting tool *groovy* in this instance—is defined.

```
<command program='${GROOVY_HOME}/bin/groovy'>
```

The actual command and any parameters it requires are passed to the tool by the `<command>` element's `<arg>` child element. Any number of `<arg>` elements can be used. The `<arg>` element has several attributes:

<arg> Element Attributes table

Attribute	Description
<code><value></code>	Specifies a parameter passed to the tool. Format is tool-specific; must be enclosed by single-quotes.
<code><path></code>	Path to files or classes required by the tool. Must be enclosed by single-quotes.
<code><file></code>	Specifies the path to any files or classes required by the tool. Format is tool-specific; must be enclosed by single-quotes.

Because `<arg>` elements are processed in the order they are defined, ensured the order conforms to that expected by the tool.

```

<command program='${GROOVY_HOME}/bin/groovy'>
  <arg value='-cp' />
  <arg path='classes:${sdkJar}:lib/commons-codec.jar:
    lib/activation-1.1.1.jar:
    lib/commons-logging.jar:lib/httpclient-cache.jar:
    lib/httpclient.jar:lib/httpcore.jar:
    lib/httpmime.jar:lib/javamail-1.4.1.jar' />

```

```
<arg file='registerInstancesWithLB.groovy' />
<arg file='${PLUGIN_INPUT_PROPS}' />
<arg file='${PLUGIN_OUTPUT_PROPS}' />
</command>
```

The `<arg file='${PLUGIN_INPUT_PROPS}' />`

specifies the location of the tool-supplied properties file.

The `<arg file='${PLUGIN_OUTPUT_PROPS}' />`

specifies the location of the file that will contain the step-generated properties.



Note: New lines are *not supported* by the `<arg>` element and are shown in this example only for presentation.

The `<post-processing>` Element

When a plug-in step's `<command>` element finishes processing, the step's mandatory `<post-processing>` element is executed. The `<post-processing>` element sets the step's output properties and provides error handling. The `<post-processing>` element can contain any valid JavaScript script (unlike the `<command>` element, `<post-processing>` scripts must be written in JavaScript). Users can also provide their own scripts when defining the step in the Serena Release Automation editor. Although not required, Serena recommends that scripts be wrapped in a `CDATA` element.

You have access to a `java.util.Properties` variable called `properties`. The `properties` variable has several special properties: `exitCode` contains the process exit code, and `Status` contains the step's status. A `Status` value of `Success` means the step completed successfully.

Another available variable— `scanner`—can scan the step's output log (scanning occurs on the agent) and take actions depending on the results. `scanner` has several public methods:

- `register(String regex, function call)` registers a function to be called when the regular expression is matched.
- `addLOI(Integer lineNumber)` adds a line to the lines of interest list, which are highlighted in the Log Viewer; implicitly called whenever `scanner` matches a line.
- `getLinesOfInterest()` returns a `java.util.List` of lines of interest; can be used to remove lines. `scan()` scans the log. Use after all regular expressions are registered.

The post-processing script can examine the step's output log, and take actions based on the result. In the following code fragment, `scanner.register()` registers a string with a regular expression engine, then takes an action if the string is found. Once all strings are registered, it calls `scanner.scan()` on the step's output log line by line.

```
![CDATA[
    properties.put("Status", "Success");
    if (properties.get("exitCode") != 0) {
        properties.put("Status", "Failure");
    }
}]
```

```

    }
    else {
        scanner.register("(?i)ERROR at line", function(lineNumber, line) {
            var errors = properties.get("Error");
            if (errors == null) {
                errors = new java.util.ArrayList();
            }
            errors.add(line);
            properties.put("Error", errors);
            properties.put("Status", "Failure");
        });
        .
        .
        .
        scanner.scan();
        var errors = properties.get("Error");
        if (errors == null) {
            errors = new java.util.ArrayList();
        }
        properties.put("Error", errors.toString());
    }
}
]]

```

You can use post-processing scripts to set output properties that can be used in other steps in the same process, which enables complex workflows. Reference prior step output properties this way:

```

${p:stepName/propName}

```

Upgrading Plug-ins

To create an upgrade:

1. Increment the number of the `version` attribute of the `<identifier>` element in `plugin.xml`.
2. Create a `<migrate>` element in `upgrade.xml` with a `to-version` attribute containing the new number.
3. Place the property and step-type elements that match the updated `plugin.xml` file within this element, as shown in the following example.

```

<?xml version="1.0" encoding="UTF-8"?>
<plugin-upgrade
    xmlns="http://www.&company;.com/UpgradeXMLSchema_v1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <migrate to-version="3">
    <migrate-command name="Run SQLPlus script">
      <migrate-properties>
        <migrate-property name="sqlFiles" old="sqlFile"/>
      </migrate-properties>
    </migrate-command>
  </migrate>

```

```
<migrate to-version="4">
  <migrate-command name="Run SQLPlus script" />
</migrate>
<migrate to-version="5">
  <migrate-command name="Run SQLPlus script" />
</migrate>
</plugin-upgrade>
```

Of course, you can also make a script-only upgrade, that is, an upgrade that contains changes to the step's associated scripts and files but does not change plugin.xml. This mechanism can be useful for plug-in development and for minor bug-fixes/updates.

The info.xml File

Use the optional info.xml file to describe the plug-in and provide release notes to users. The file's `<release-version>` element can be used for version releases.