SERENA®

RELEASE AUTOMATION

User's Guide

Serena Proprietary and Confidential Information

Trademarks

U.S. Government Rights

Part number: Serena ALM Product version: 4.5.1

Publication date: 2013-07-30

Table of Contents

Part 1: Introduction 17

Chapter 1: Welcome to Serena Release Automation 19

About This Documentation 19

Serena Support 19

Overview 19

Chapter 2: Elements of Serena Release Automation 21

Components 21

Component Processes 22

Plug-ins 23

Component Versions and the CodeStation Repository 24

Applications 24

Application Processes 24

Snapshots 25

Environments 25

Resources 25

Resource Groups 26

Agents 26

Chapter 3: User Interface Overview 27

Home Page 27

Management Page 28

Administration Page 28

Part 2: Installation and Configuration 29

Chapter 4: Architecture Overview 31

Services 32

Agents 36

Server-Agent Communication 37

Agent Relays: Crossing Network Boundaries and Firewalls 38

Chapter 5: Installing Servers and Agents 41

Installation Checklist 41

Installation Recommendations 42

System Requirements 43

Server Minimum Installation Requirements 43

Server Installation Recommendations 43

Agent Minimum Requirements 44

Database Requirements 44

32- and 64-bit JVM Support 44

Database Installation 45

Installing Oracle 46

Installing MySQL 47

Installing Microsoft SQL Server 47

Server Installation 49

Downloading Serena Release Automation 49

Interactive Server Installation (Windows, Linux/UNIX (AIX, Solaris) 49

(Windows) Server Install: Destination Folder Panel 50

(Windows) Server Install: Administrator Details Panel 51

Server Installation (Other UNIX Platforms) 51

Silent Mode Server Installation 52

(Windows) Server Silent Installation 52

(Windows) Server Silent Install Options 52

(Windows) Server Silent Install: OptionsFile.txt Examples 54

(Linux/UNIX Platforms) Server Silent Install 56

(Linux/UNIX) Server Silent Install Options 56

(Linux/UNIX) Server Silent Install: optionsFile.txt Examples 59

Single Sign On (SSO) with Serena Business Manager (SBM) 60

Integrating with Serena Business Manager using SSO 61

Configuring Tomcat for SSO 61

Single Sign Out 63

Agent Installation 63

Interactive Agent Installation (Windows, Linux/UNIX (AIX, Solaris, HP-UX) 64

Silent Mode Agent Installation 64

(Windows) Agent Silent Installation 64
(Windows) Agent Silent Install Options 64
(Windows) Agent Silent Install: OptionsFile.txt Example 66
(Linux/UNIX Platforms) Agent Silent Installation 67
(Linux/UNIX) Agent Silent Install Options........ 67
(Linux/UNIX) Agent Silent Install: OptionsFile.txt Examples 68
Installing Agent Relays 69
Chapter 6: Running Serena Release Automation 73
Stopping and Starting the Server 73
Starting and Stopping an Agent Relay 73
Starting and Stopping an Agent 74
Accessing Serena Release Automation 74
Chapter 7: Server Configuration........ 75
Multiple-server High Availability Configuration........ 75
Active-active Server Installation 75
Connecting Agents to a Single Endpoint 76
Connecting Agents to a Series of Server Endpoints 76
Two-server Cold Standby Configuration........ 77
Typical Data Center Configurations 77
Chapter 8: SSL Configuration 81
Configuring Unauthenticated Mode for HTTP Communication 81
Configuring Mutual Authentication Mode 82
Property Settings for Mutual Authentication 82
Adding an Alias to an Agent 83
Adding an Alias to an Agent Relay 84
Mutual Authentication: Server and Agent(s) 84
Mutual Authentication: Server, Agent Relay, and Agent(s) 85
Chapter 9: Automation Administration 89
Automation Plug-ins 89
Locks 89
Managing Locks 90

Post-Processing Scripts 91

Statuses 91

Resources Inventory 91

Component Inventory 91

Environment Inventory 92

Chapter 10: Security Administration 93

Setting up Security 93

Roles and Permissions 94

Default Roles 95

Creating and Editing Roles 95

Agent Roles 96

Application Roles 97

Component Template Roles 97

Component Roles 97

Environment Roles 98

License Roles 98

Resource Roles 99

System Security 99

User Interface Security 100

Authorization Realms 101

Creating an LDAP Authorization Realm 101

Creating Authorization Groups 102

Default Permissions 103

Setting Default Permissions 103

Authentication Realms 104

Creating an Authentication Realm 104

Creating an LDAP Authentication Realm 105

Authentication Realm Users 106

Importing LDAP Users 106

To Import LDAP Users 106

Tokens 106

Chapter 11: System Administration 109
Licenses 109
Adding a License 109
Adding Agents to a License 110
Modifying or Deleting a License 110
Network Relay 110
Notifications 111
Creating Notification Templates 112
Output Log 113
System Settings 113
Chapter 12: File Versioning Repository 115
Codestation 115
Relocating Codestation 116
Chapter 13: Integrating Release Automation with Serena Solutions 117
Part 3: Using Serena Release Automation 119
Chapter 14: Managing Components 121
Component Creation Overview 121
Creating Components 122
Basic Fields 122
Source Configuration Fields 124
File System (Basic) 125
File System (Versioned) 126
Serena Dimensions CM 126
Serena PVCS 127
Component Properties 128
Importing/Exporting Components 129
Exporting Components 130
Importing Components 130
Component Versions 131
Importing Versions Manually 131
Importing Versions Automatically 133

Component Version Statuses 133
Deleting Component Versions 133
Inactivating Component Versions 133
Component Change Logs 133
Component Processes 134
Configuring Component Processes 134
Process Editor 135
Displaying the Process Editor 136
Using the Process Editor 136
Adding Process Steps 138
Connecting Process Steps 140
Process Properties 142
Switch Steps and Conditional Processes 143
Configuring Access for Process Steps 145
User Impersonation 145
User Impersonation on UNIX/Linux Using sudo (or su) 145
Impersonation on Windows Systems 146
Component Manual Tasks 147
Creating Component Manual Tasks 147
Using Component Manual Tasks 147
Post-Processes 148
Component Templates 148
Creating a Component Template 149
Importing/Exporting Templates 149
Exporting Templates 149
Importing Templates 150
Component Template Properties 150
Using Component Templates 152
Configuration Templates 153
Deleting and Deactivating Components 153
Chapter 15: Managing Applications 155

Application Creation Overview 155
Creating Applications 156
Adding Components to an Application 157
Adding Application Configuration Properties 157
Modifying and Deleting Application Configuration Properties........................... 158
Importing/Exporting Applications 158
Exporting Applications 158
Importing Applications 158
Application Environments 160
Creating an Environment 160
Mapping Resources to an Environment 161
Environment Properties 161
Application Processes 162
Creating Application Processes 162
Application Process Step Details 163
Finish 163
Install Component 163
Uninstall Component 164
Rollback Component 165
Manual Application Task (Utility) 166
Application Manual Tasks 167
Creating Application Manual Tasks 167
Using Manual Tasks 167
Approval Process 167
Work Items 168
Snapshots 168
Creating Snapshots 169
Snapshot Options 169
Application Gates 170
Creating Gates 171
Structure of the default.xml File 171

Chapter 16: Managing Resources 173
Resource Groups 173
Creating a Resource Group 173
Resource Roles 174
Role Properties 174
Agents 174
Remote Agent Installation 175
Managing Agents Remotely 176
Agent Pools 177
Creating an Agent Pool 177
Managing Agent Pools 177
Chapter 17: Configuration and Processes Options 179
Application/Component/Environment Configuration 179
Standalone Processes 179
Chapter 18: Deployments 181
Deploying Components 182
Scheduling Deployments 183
Chapter 19: Reports 185
Deployment Reports 186
Deployment Detail Report 186
Deployment Detail Fields 186
Running the Deployment Detail Report 187
Report Samples: Deployment Detail 188
Deployment Count Report 189
Deployment Count fields 190
Running the Deployment Count Report 190
Report Samples: Deployment Count 192
Deployment Average Duration Report 193
Deployment Average Duration Fields 194
Running the Deployment Average Duration Report 194
Sample Reports: Deployment Average Duration 195

Deployment Total Duration Report 196
Deployment Total Duration Fields 197
Running the Deployment Total Duration Report.. 197
Sample Reports: Deployment Total Duration 198
Security Reports 199
Application Security Report 199
Application Security Fields 199
Component Security Report 200
Component Security Fields 200
Environment Security Report.. 201
Environment Security Fields 201
Resource Security Report 201
Resource Security Fields 201
Saving and Printing Reports 202
Saving Report Data 202
Saving Report Filters.. 202
Printing Reports 203
Chapter 20: Example—helloWorld Deployment 205
helloWorld: Creating Components 205
helloWorld Deployment 206
helloWorld: A Note Before You Begin 206
helloWorld: Component Version 206
helloWorld: Component Process 209
helloWorld: Process Design 210
helloWorld: Application 213
helloWorld: Creating an Application.. 214
Adding the helloWorld Component to the Application 214
helloWorld: Adding an Environment to the Application 214
helloWorld: Adding a Process to the Application 216
Designing the Process Steps 216
Running the Application 218

Chapter 21: Serena Release Automation Properties 219

Chapter 22: Command Line Client (CLI) 223

Command Format 223

Commands 224

addActionToRoleForApplications 224

addActionToRoleForComponents 224

addActionToRoleForEnvironments 225

addActionToRoleForResources 225

addActionToRoleForUI 225

addComponentToApplication 226

addGroupToRoleForApplication 226

addGroupToRoleForComponent 226

addGroupToRoleForEnvironment 227

addGroupToRoleForResource 227

addGroupToRoleForUI 228

addLicense 228

addNameConditionToGroup 228

addPropertyConditionToGroup 229

addResourceToGroup 229

addRoleToResource 230

addRoleToResourceWithProperties 230

addUserToGroup 230

addUserToRoleForApplication 231

addUserToRoleForComponent 231

addUserToRoleForEnvironment 232

addUserToRoleForResource 232

addUserToRoleForUI 232

addVersionFiles 233

addVersionStatus 233

createApplication 234

createApplicationProcess 234
createComponent 235
createComponentProcess 235
createDynamicResourceGroup 235
createEnvironment 236
createGroup 236
createMapping 237
createResourceGroup 237
createRoleForApplications 237
createRoleForComponents 238
createRoleForEnvironments 238
createRoleForResources 238
createRoleForUI 239
createSubresource 239
createUser 239
createVersion 240
deleteGroup 240
deleteResourceGroup 240
deleteResourceProperty 241
deleteUser 241
exportGroup 241
getApplication 241
getApplicationProcess 242
getApplicationProcessRequestStatus 242
getApplications 242
getComponent 243
getComponentProcess 243
getComponents 243
getComponentsInApplication 243

getEnvironment 244
getEnvironmentsInApplication 244
getMapping 244
getResource 245
getResourceGroup 245
getResourceGroups 245
getResourcesInGroup 245
getResources 246
getResourceProperty 246
getRoleForApplications 246
getRoleForComponents 246
getRoleForEnvironments 247
getRoleForResources 247
getRoleForUI 247
getUser 248
importGroup 248
importVersions 248
login 249
logout 249
removeActionFromRoleForApplications 249
removeActionFromRoleForComponents 249
removeActionFromRoleForEnvironments 250
removeActionFromRoleForResources 250
removeActionFromRoleForUI 250
removeGroupFromRoleForApplication 251
removeGroupFromRoleForComponent 251
removeGroupFromRoleForEnvironment 252
removeGroupFromRoleForResource 252
removeGroupFromRoleForUI 252

removeResourceFromGroup 253
removeRoleForApplications.. 253
removeRoleForComponents 254
removeRoleForEnvironments 254
removeRoleForResources 254
removeRoleForUI 254
removeRoleFromResource 254
removeUserFromGroup 255
removeUserFromRoleForApplication.. 255
removeUserFromRoleForComponent 255
removeUserFromRoleForEnvironment 256
removeUserFromRoleForResource 256
removeUserFromRoleForUI.. 257
repeatApplicationProcessRequest 257
requestApplicationProcess 257
setComponentEnvironmentProperty.. 258
setComponentProperty 258
setResourceProperty.. 259
updateUser 259

Part 1: Introduction

This section contains the following information:

- Chapter 1: Welcome to Serena Release Automation [page 19]
- Chapter 2: Elements of Serena Release Automation [page 21]
- Chapter 3: User Interface Overview [page 27]

Chapter 1: Welcome to Serena Release Automation

Serena Release Automation enables you to automate the deployment of application changes. Benefits include continuous delivery and DevOps automation, reduction of development costs, and increased deployment frequency without increased risk.

About This Documentation

This documentation guides you through installing and using the Serena Release Automation product and is intended for all users.

This documentation is available in PDF and HTML formats.

Serena Support

The Serena Support website, http://support.serena.com, provides additional information about Serena products, including a searchable Knowledgebase.

Note: You need an account to log in to the Serena Support website. If you do not have an account, contact your Sales Representative.

Overview

Software deployment is the process of moving software through various pre-production stages to final production. Typically, each stage represents a step of higher criticality, such as quality assurance to production.

Software deployment complexity increases with more releases to deploy, more deployment targets, more types of deployment targets, shortened deployment cycles, and changes in technology.

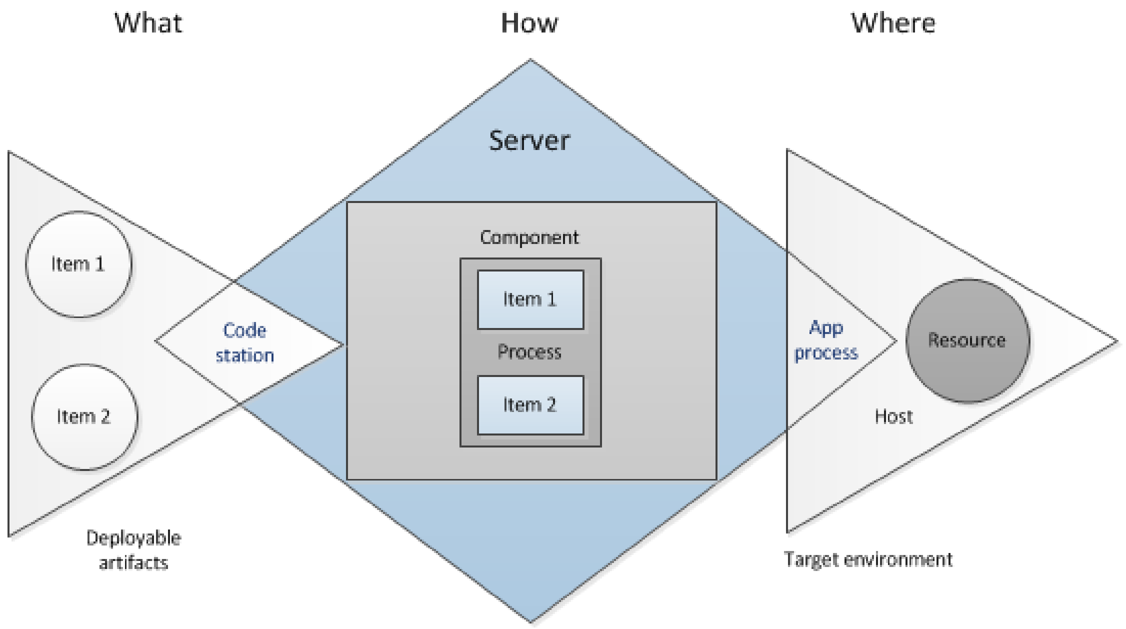
Serena Release Automation helps you meet the deployment challenge by providing tools that improve deployment speeds while simultaneously improving their reliability.

With Serena Release Automation, you can:

- Model processes that orchestrate complex deployments across every environment and approval gate with complete visibility into *n*-tiered deployments
- Visualize the end-to-end deployment process and develop the big picture, the *What*, *How*, and *Where* of the deployment workflow, using drag-and-drop design tools
  - **What**: the deployable items that Serena Release Automation delivers to target destinations: binaries, static content, middleware updates, database changes and configurations, and anything else associated with the software.
  - **How**: by combining deployable items with processes to create components, and designing applications that coordinate and orchestrate multi-component deployments.

- **Where:** the target destination's hosts and environments

Figure 1. Deployment Process



Chapter 2: Elements of Serena Release Automation

Serena Release Automation elements include the following:

Property	Description
Components [page 21]	Components are the building blocks of Serena Release Automation. They represent deployable items and have user-defined *component processes* that operate on those items. After defining a component's source and processes, you import its artifacts into Serena Release Automation's artifact repository, CodeStation, to store *component versions*.
Applications [page 24]	Applications initiate component deployments; they bring together components with their deployment targets, and orchestrate multi-component deployments. When you create an application, you identify the included components and define an *application process*. You can create a *snapshot* to capture the application's current state, and as the application moves through different environments, the snapshot ensures that proper component versions are used.
Environments [page 25]	An environment is a user-defined collection of resources that host an application. A resource can represent a physical machine or a specific target on a machine, such as a database or server. A host machine can have several resources represented on it, and a resource can represent a process distributed over several physical or virtual machines. An agent is a process that runs on target host and communicates with the Serena Release Automation server. Typically, an agent runs on the same host where the resources it handles are located. A single agent can handle all resources on its host.

Components

Components represent deployable items along with user-defined processes that operate on them, usually by deploying them. Deployable items, also called artifacts, can be files, images, databases, configuration materials, or anything else associated with a software project. Components have *versions* which are used to ensure that proper component instances get deployed.

Artifacts can come from a number of sources: file systems, build servers such as AnthillPro, source version control systems, Maven repositories, as well as many others. When you create a component, you identify the source and define how the artifacts will be brought into Serena Release Automation. If the source is Subversion, for example, you specify the Subversion repository containing the artifacts. Each component represents artifacts from a single source.

Component Processes

A component process is a series of user-defined steps that operate on a component's artifacts. Each component has at least one process defined for it and can have several.

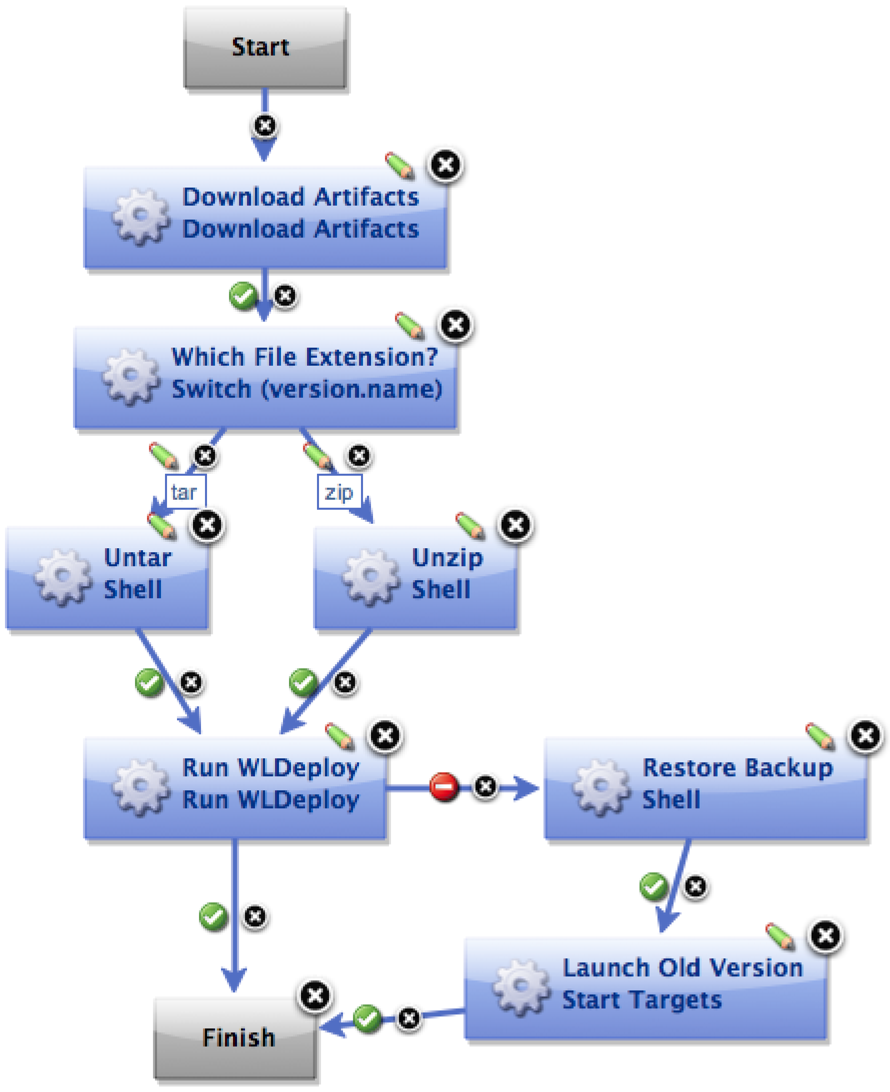
Component processes are executed by Serena Release Automation agents running on hosts. One instance of a component process is invoked for each resource mapped to a component in the target environment.

You create component processes using Serena Release Automation's process editor. The *process editor* is a visual drag-and-drop editor that enables you to drag process steps from a menu of standard steps onto the *design space* and configure them immediately. As you place additional steps, you visually define their relationships with one another.

Component processes may have only one step or may have many steps and relationships. They may also have conditional steps, or switch steps. When you initially implement Serena Release Automation, you will typically use these process steps to replace existing deployment scripts and manual processes.

You can choose from several default utility processes, such as inventory management and workflow control, and you can select from a list of over 60 provided plug-ins, which provide support for many common processes, such as downloading and uploading artifacts and retrieving environment information.

Figure 1. Process Editor with a Component Process



Plug-ins

Plug-ins provide basic processing functions as well as integration with third-party tools. Serena Release Automation ships with plug-ins for many common deployment processes, and others are readily available for a wide variety of tools, such as middleware tools, databases, servers, and other deployment targets.

Plug-ins break down a tool's functions into simple, discrete steps such as detecting if a server is stopped, stopping a server, and starting a server.

When you use plug-ins to create a component process, you can use steps from several plug-ins. For example, you might create a process using a plug-in for a source control tool that deploys a component to a middleware server, and another plug-in to configure a step that removes the component from the server.

For a reference of plug-ins and examples of component processes using plug-ins, see the *Serena Release Automation Plug-ins Guide*.

Component Versions and the CodeStation Repository

After defining a component's source and processes, you import its artifacts into Serena Release Automation's artifact repository CodeStation. Artifacts can be imported automatically or manually.

By default, a complete copy of an artifact's content is imported into CodeStation. The original artifacts are untouched. This provides tamper-proof storage and the ability to review and validate artifacts with the Serena Release Automation user interface. If you have storage concerns or use a tool like Maven, you can limit CodeStation to using references to the artifacts instead of actually copying them.

Each time a component is imported, including the first time, it is versioned. Versions can be assigned automatically by Serena Release Automation, applied manually, or come from a build server. Every time a component's artifacts are modified and re-imported, a new version of the component is created.

A component may have several unique versions in CodeStation. A version can be full or incremental. A full version contains all component artifacts; an incremental version only contains artifacts modified since the previous version was created.

Applications

Applications initiate component deployments. They bring together components with their deployment targets and orchestrate multi-component deployments.

Application Processes

When you create an application, you identify the included components and define an application process. Application processes, like component processes, are created with the process editor. Serena Release Automation provides several common process steps. Otherwise, application processes are configured from processes defined for their associated components.

Application processes can run manually, automatically on some trigger condition, or on a user-defined schedule. When a component has several processes defined for it, the application determines which ones are executed and in which order.

For example, an *n*-tiered application might have a web tier, a middleware tier, and a database tier. The database tier must be updated before the other two, which are then deployed concurrently. An application can orchestrate the entire process, including putting servers on- and off-line for load-balancing as required.

When an application process executes, it interacts with a specific environment. An environment is a collection of one or more resources. At least one environment must be associated with the application before the process can be executed.

Application processes are environment agnostic; processes can be designed independently of any particular environment. This enables a single application to interact with separate environments, such as QA, or production. To use the same application process with multiple environments, you associate each environment with the application and execute the process separately for each one.

In addition to deployments, several other common processes are available, such as rolling back deployments. Serena Release Automation tracks the history of each component version, which enables application processes to restore environments to any desired point.

Snapshots

A snapshot is a collection of specific component versions, usually versions that are known to work together. A snapshot captures the application's current state, and as the application moves through different environments, the snapshot ensures that proper component versions are used.

Snapshots help manage complex deployments, such as deployments with multiple tiers and development teams. For example, after testing and confirming that team A's component works with teams B's, a snapshot can be taken. Then, as development progresses, additional snapshots can be taken and used to model the effort and drive the entire deployment, coordinating versions, configurations, and processes.

Typically, a snapshot is generated in an uncontrolled environment, meaning one without approvals.

Environments

An environment is a user-defined collection of resources that host an application. Environments are typically modeled on some stage of the software project life cycle, such as development, QA, or production. A resource is a deployment target, such as a database or J2EE container. Resources are typically found on the same host where the agent that manages them is located. A host can be a physical machine, virtual machine, or be cloud-based.

Environments can have different topologies. For example, an environment can consist of a single machine, be spread over several machines, or be spread over clusters of machines.

Environments are application-scoped. Although multi-tenant machines can be the target of multiple applications, IT organizations often use application-specific environments. Additionally, approvals are typically scoped to environments.

Serena Release Automation maintains an inventory of every artifact deployed to each environment and tracks the differences between them.

Resources

A resource represents a deployment target, such as a physical machine, virtual machine, database, or J2EE container. Components are deployed to resources by agents, which are physical processes. Resources generally reside on the same host where its managing agent runs. A host can have more than one resource. If an agent is configured to handle multiple resources, a separate agent process is invoked for each one.

A resource can represent a physical machine, which is the simplest configuration, or a specific target on a machine, such as a database or server. A host machine can have several resources represented on it. In addition, a resource can represent a process distributed over several physical or virtual machines.

To perform a deployment, at least one resource must be defined and for non-trivial deployments, at least one agent must be defined. Typically, each host in a participating environment has an agent running on it to handle the resources located there.

A proxy resource is a resource effected by an agent on a host other than the one where the resource is located. If an agent does not require direct interaction with the file system or with process management on the host, a proxy resource can be used. When a deployment needs to interact with a service exposed on the network, such as a database or J2EE server, the interaction can happen from any machine that has access to the networked service.

Resource Groups

A resource group is a logical collection of resources. Resource groups enable collections of resources to be easily reused. Resource groups can manage multi-tenant scenarios, for example, in which several machines share the same resources.

Agents

An agent is a process that runs on target host and communicates with the Serena Release Automation server. Agents are integral to Serena Release Automation's client/server architecture. Agents perform the actual work of deploying components and so relieves the server from the task, making large deployments involving thousands of targets possible.

Typically, an agent runs on the host where the resources it handles are located. A single agent can handle all resources on its host. If a host has several resources, an agent process is invoked separately for each resource.

For example, a test environment might contain a single web server, a single middleware server, and a single database server all running on the same host machine. A deployment to this environment might have one agent and three separate resources.

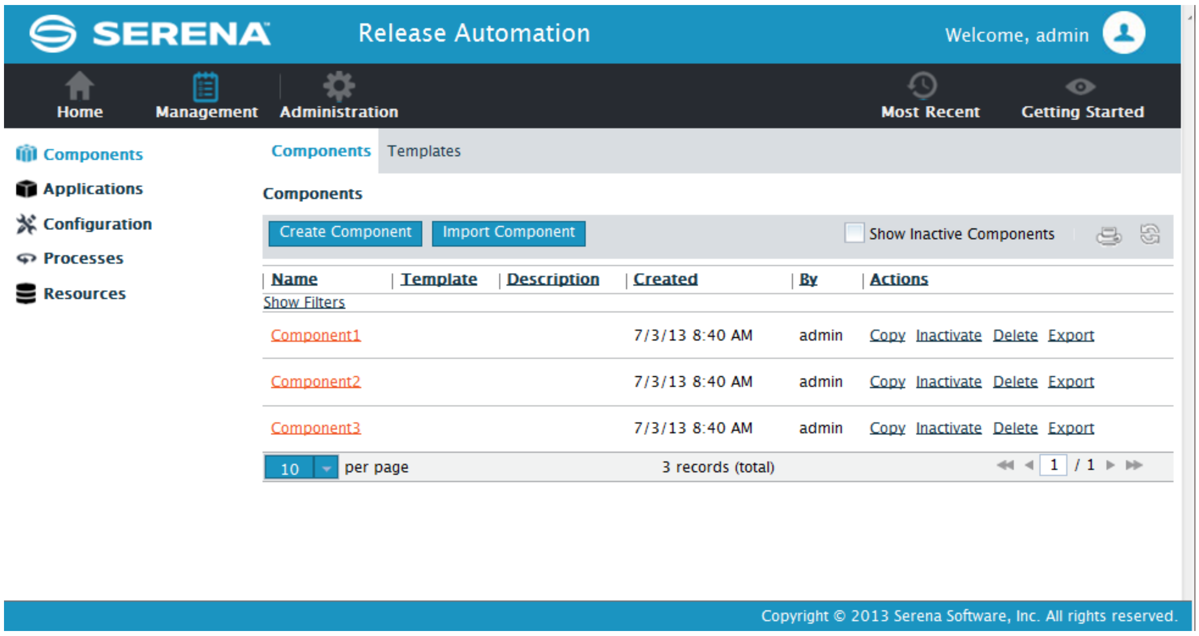
Depending on the number of hosts in an environment, a deployment might require a large number of agents. Agents are unobtrusive and secure.

Agent communications use SSL encryption and mutual key-based authentication. For added security, agents do not listen to ports, but instead open direct connections to the server.

Chapter 3: User Interface Overview

The Serena Release Automation Web client user interface enables you to create and configure the Release Automation elements and initiate the deployment of your components.

Figure 1. User Interface Elements



Name	Template	Description	Created	By	Actions
Component1			7/3/13 8:40 AM	admin	Copy Inactivate Delete Export
Component2			7/3/13 8:40 AM	admin	Copy Inactivate Delete Export
Component3			7/3/13 8:40 AM	admin	Copy Inactivate Delete Export

10 per page   3 records (total)   1 / 1

- Home Page [page 27]
- Management Page [page 28]
- Administration Page [page 28]

Home Page

The Home page of the user interface appears when you select the Home button.

Left Navigation Option	Includes Options Such As
Dashboard	View current activity
Calendar	View the process schedule
Work Items	View your work items

Left Navigation Option	Includes Options Such As
Reports	View provided reports on deployment and security and additional reports you have created

Management Page

The Management page of the user interface appears when you select the Management button.

Left Navigation Option	Includes Options Such As
Components	Create and import components and component templates
Applications	Create and import applications
Configurations	Configure applications, components, and environments in context of the hierarchy
Processes	Create and import standalone processes
Resources	Create resources

Administration Page

The Administration page of the user interface appears when you select the Administration button.

Left Navigation Option	Includes Options Such As
Automation	Load plug-ins, view locks, configure post-processing scripts, set statuses
Security	Configure users, groups, roles, and permissions and system and UI security
System	Enter licenses, configure notifications, add properties, and configure system settings

Part 2: Installation and Configuration

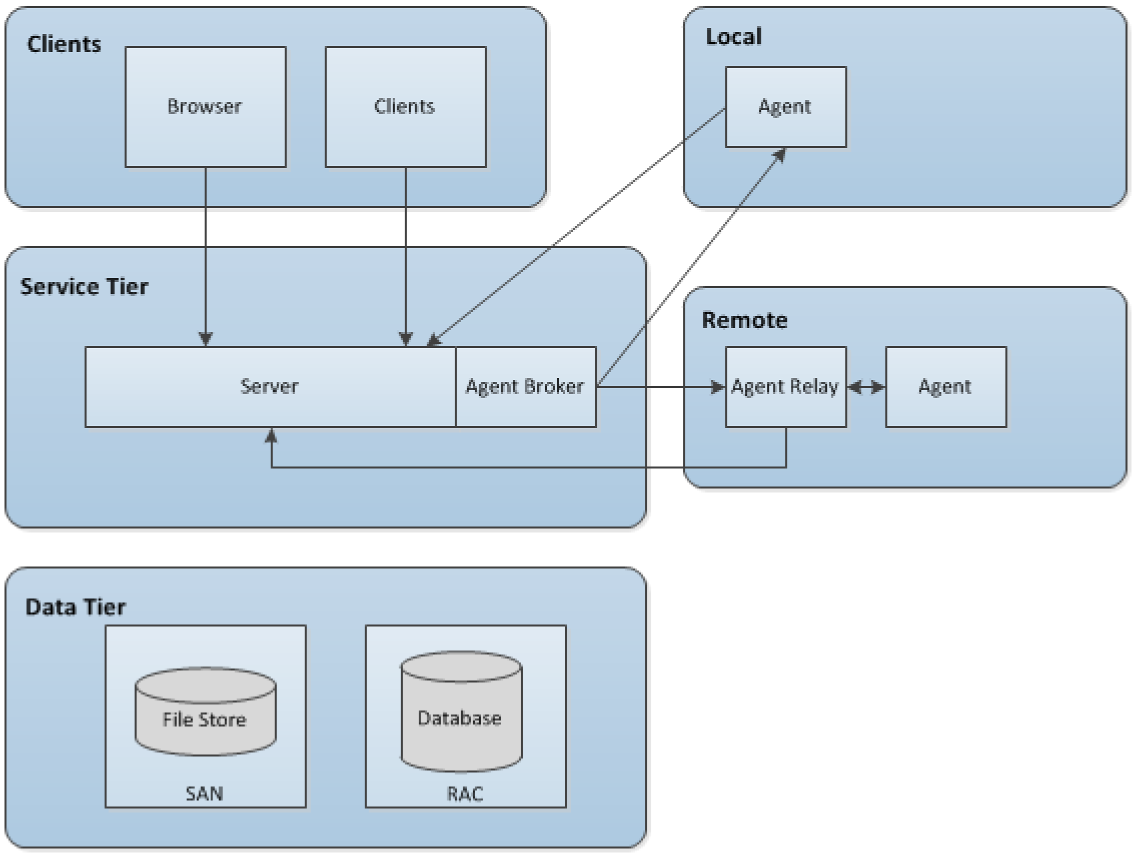
This section contains the following information:

- Chapter 4: Architecture Overview [page 31]
- Chapter 5: Installing Servers and Agents [page 41]
- Chapter 6: Running Serena Release Automation [page 73]
- Chapter 7: Server Configuration [page 75]
- Chapter 8: SSL Configuration [page 81]
- Chapter 9: Automation Administration [page 89]
- Chapter 10: Security Administration [page 93]
- Chapter 11: System Administration [page 109]
- Chapter 12: File Versioning Repository [page 115]
- Chapter 13: Integrating Release Automation with Serena Solutions [page 117]

Chapter 4: Architecture Overview

Serena Release Automation architecture consists of clients, a service tier, and a data tier.

Figure 1. Serena Release Automation Architecture



Clients

Serena Release Automation clients include the following:

- *Web browser*: a Rich Internet Application (RIA) that maintains much of its functionality in the browser
- *Command line*: provides most of features found in the browser-based GUI

Clients:

- communicate with the server via HTTP or HTTPS
- may be deployed locally, on the same LAN as the Serena Release Automation server
- may be deployed remotely

- interact with RESTful (representational state transfer) services on the server as needed

Service Tier

The service tier has a central server that provides a web server front-end and core services, such as workflow, agent management, deployment, inventory, and security.

A service can be thought of as a self-contained mechanism for hosting a piece of business logic. Services can be consumed by clients\agents or other services.

Deployments are orchestrated by the server and performed by agents distributed throughout the network. Most clients use browsers to communicate with the web server via HTTP(S). Most server-agent communication is done via JMS (discussed below) but HTTP(S) is also used as required.

Serena Release Automation uses stateless communications for server-agent communications (JMS-based) as well as client-web service communications. Stateless, as used here, means the server retains little session information between requests and each request contains all the information required to handle it. The server sets-up listening sockets and listens for agents, relays, and users (clients).

For added security, agents do not listen on ports. Agents send requests when they are ready to make the transition to a new state.

Server-agent communication is built around transferring—deploying—components. Components can contain any business-meaningful content, such as environment information, configuration data, source, static files, or anything else associated with a software project.

Because JMS connections are persistent and not based on a request-response protocol, Serena Release Automation does not have to continually open and close ports, which enables the server to communicate with agents at any time while remaining secure and scalable.

Many Serena Release Automation services are REST-type (representational state transfer). REST-style services are web services that focus on transferring resources over HTTP.

A resource can be any business-meaningful piece of data. Resources are transferred by a self-describing format such as XML or JSON (JavaScript Object Notation). The XML and JSON representations typically model resource states at the time of agent/client requests.

REST-style services achieve statelessness by ensuring that requests include all the data needed by the server to make a coherent response.

Data Tier

The data tier's relational database stores configuration and run-time data. The data tier's file store—CodeStation—contains log files, artifacts, and other non-structured data objects. Reporting tools can connect directly to the relational database.

Services

The Serena Release Automation server provides a variety of services, such as:

- the user interface
- component and application configuration tools

- workflow engine
- security services

Workflow requests are initiated with either the web client user interface or the CLI (command line interface). When a workflow is requested, many services are used to fulfill the request.

Figure 1. Services and Process Workflow

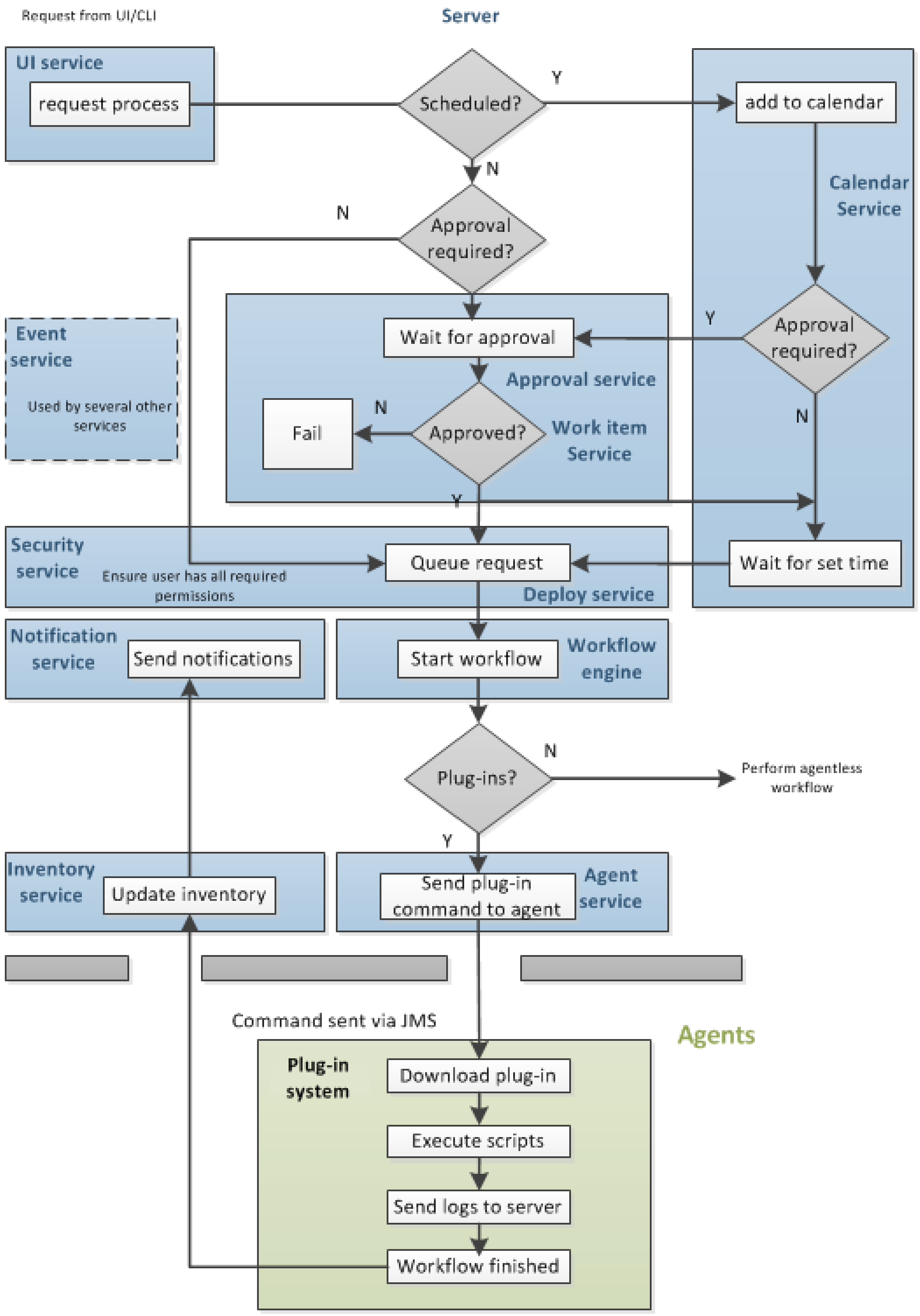


Table 1. Services Table

Service	Description
User Interface	Used to create components and fashion workflows, request processes and manage security and resources, among other things. REST-based.
Workflow Engine	Manages workflows—application and component processes. Calls the agent responsible for performing the workflow's current plug-in step. When the workflow is finished, alerts the notification and inventory services. Called by the deploy service. REST-based.
Agent	Tracks installed agents and routes plug-in commands to affected agents. Commands come from plug-in steps. Invoked by the workflow service. REST-based.
Work Item	Operates in tandem with the approval service; provides approver alerts and enables approvers to accept or reject workflows. If a scheduled workflow remains unapproved at run-time, the job fails automatically. REST-based.
Plug-in Manager	Serena Release Automation can interact with virtually any system through its extensible plug-in system; plug-ins provide functions by breaking-down tool features into automated steps. Plug-ins can be configured at design- and run-time. When a plug-in step executes, the controlling agent invokes its run-time process to execute the step. When a new component version is available, the agent compares the current component version and downloads and only new or changed artifacts.
Event	The event service is ubiquitous; it alerts other services as various trigger conditions occur.
Deployment Service	Manages deployments. When a deployment process is requested, invokes the workflow engine to perform the process. Works in tandem with the security service to ensure users have required permissions. REST-based.
Notification Manager	Notifies users about the status of deployments; notifications are sent to approvers if the system is configured with an email server and the user has an email address. Invoked by the workflow manager. REST-based.
Inventory Manager	When a workflow finishes, the inventory manager updates affected inventory records. Serena Release Automation maintains an inventory of every deployed artifact in every environment, which provides complete visibility across environments. REST-type service.
Approval Engine	Enables creation of approval-requiring jobs and approver roles. Works in tandem with the work item service to ensure required approvals are made before scheduled jobs. REST-based.

Service	Description
Security	Controls what users can do and see; maps to organizational structures by teams, roles, activities, etc. REST-based.
Calendar	Used to schedule processes to being at some future point; works in tandem with the approval and work item services. REST-based.
CodeStation	Handles versioning of artifacts; agents invoke it when downloading component versions. REST-based.

Agents

Agents play a central role in the Serena Release Automation architecture. An agent is a lightweight process that runs on a deployment-target host and communicates with the Serena Release Automation server.

Agents perform the actual work of deployment which relieves the server from the task. All processes—packaging, configuration, deployments, and so on—requested by the Serena Release Automation server are executed on hardware assigned to agents.

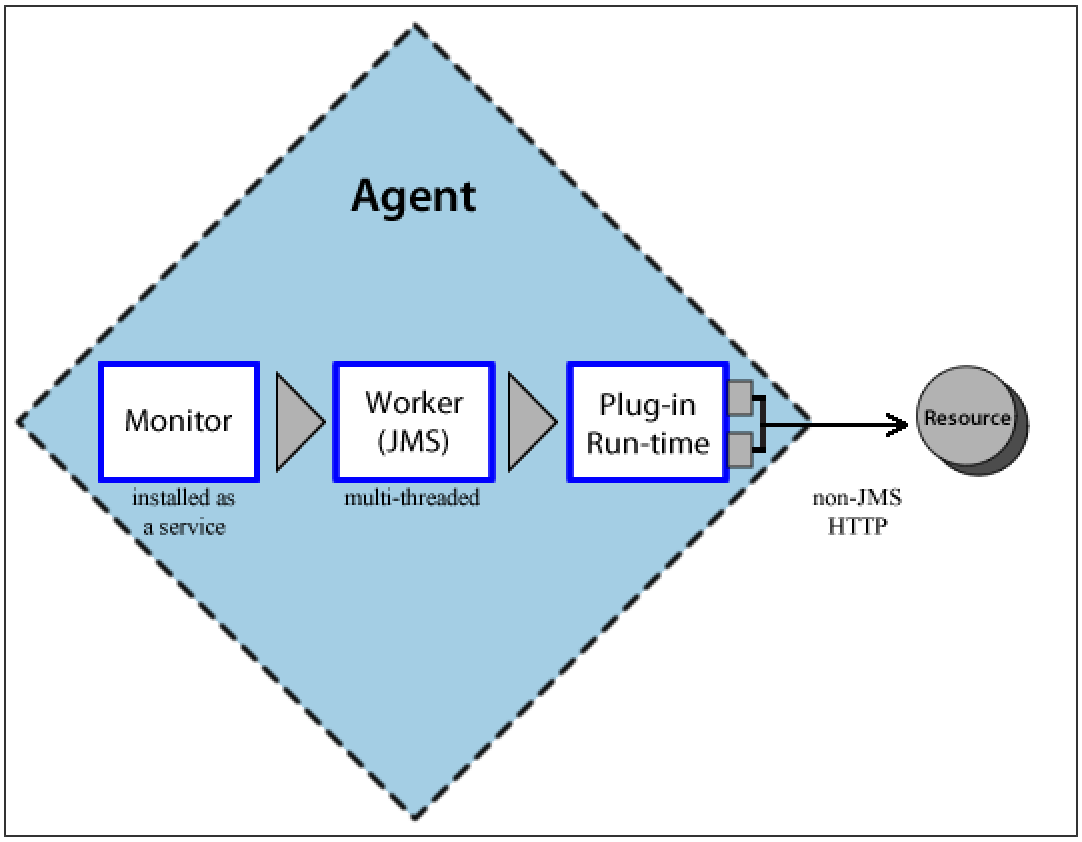
Once an installed agent has been started, the agent opens a socket connection to the Serena Release Automation server. Communication between server and agents uses a JMS-based (Java Message Service) protocol and can be secured using SSL, with optional mutual key-based authentication for each end-point. This communication protocol is stateless and resilient to network outages (the benefits of statelessness are discussed below).

While we characterize an agent as a single process, technically an agent consists of a *worker* process and a *monitor* process. The worker is a multi-threaded process that performs the actual deployment work after receiving commands from the server.

Work commands come from plug-in steps which provide seamless integration with many third-party tools. The monitor is a service that manages the worker process–starting and stopping, handling restarts, upgrades, and security, for example.

Agents are rarely upgraded because their functionality is derived from plug-ins, which can be upgraded at will. Once an agent is installed, it can be managed from the Serena Release Automation web application.

Figure 1. Agent Processes



Agents are an important part of Serena Release Automation's scalability. By adding more agents, the throughput and capacity of the system increases almost exponentially and so can scale to fit even the largest enterprise.

Server-Agent Communication

Most agent communication is done with JMS, but some agent activities—posting logs, transmitting test results, or posting files to Codestation, for example—use the web tier via HTTP(s) as needed.

The JMS channel is Serena Release Automation's primary control channel; it's the channel the server uses to send agent commands.

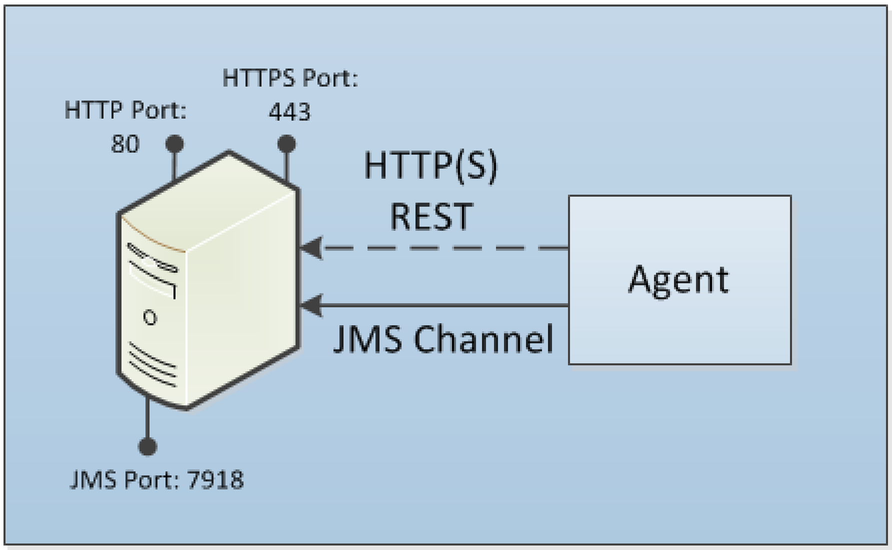
By default the server listens on only three ports: port 7918 for JMS, 8080 for HTTP, 8443 for HTTPS.

The agent monitor service uses JMS for all server communications and for sending commands, such as "run step," to the worker process. The worker process uses JMS for system communications, and HTTP REST services when performing plug-in steps or retrieving information from the server.

Stateless server-agent communication provides significant benefits to performance, security, availability, and disaster recovery. Because each agent request is self-contained, a transaction consists of independent message which can be synchronized to secondary storage as it occurs. Either endpoint–server or agent∇can be taken down and brought back up without repercussion (other than lost time). If communications fail mid-transaction, no messages are lost.

Once reconnected, the server and agent automatically determine which messages got through and what work was successfully completed. After an outage, the system synchronizes the endpoints and recovers affected processes. The results of any work performed by an agent during the outage are communicated to the server.

Figure 1. Stateless Communication



In the Server-Agent Communication [page 37] figure, the arrow represents the direction in which communication was established, but the flow can be in both directions with JMS.

Agent Relays: Crossing Network Boundaries and Firewalls

Serena Release Automation supports remote agents cross-network deployments. As long as there is at least a low bandwidth WAN connection between the server and remote agents, the Serena Release Automation server can send work to agents located in other geographic locations.

To aid performance and ease maintenance, Serena Release Automation uses agent relays to communicate with remote agents. An agent relay requires that only a single machine in the remote network contact the server.

Other remote agents communicate with the server by using the agent relay. All agent-server communication from the remote network goes through the relay. Agent relays can be configured as CodeStation proxies in order to optimize the transfer of large objects.

The following, a simple artifact move, illustrates the mechanics of remote communications:

1. A remote agent starts and establishes a connection to the agent relay via JMS, which, in turn, alerts the Serena Release Automation server via JMS that the remote agent is online.

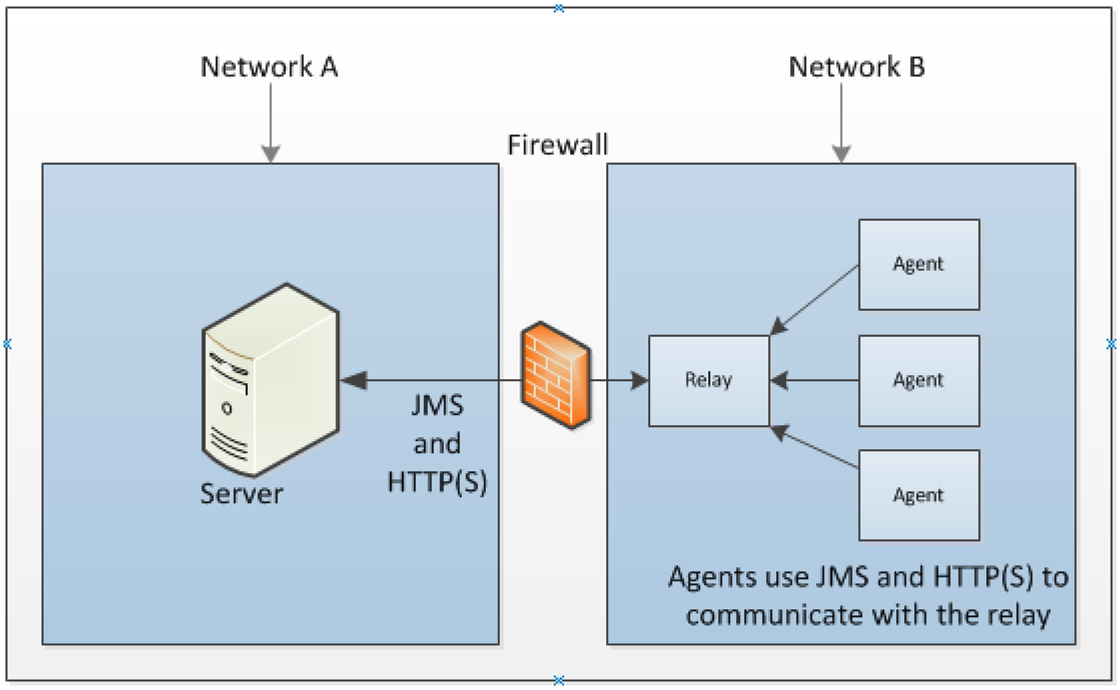
2. The server sends, say, an artifact download command to the relay via JMS, and the relay delivers the message to the remote agent (also via JMS).

3. The server locates the artifacts, and then:

a. Uploads the artifacts to the relay over HTTP(s), which begins streaming them directly to the agent over the server-relay HTTP(s) connection.

b. Once the remote agent completes the work, it informs the server via JMS.

Figure 1. Crossing Network Boundaries



By default, agent relays open the connection to the Serena Release Automation server, but the direction can be reversed if your firewall requires it. Remote agents open connections to the agent relay.

In configurations with relay agents, agents local to the Serena Release Automation server continue to use direct communications.

Chapter 5: Installing Servers and Agents

The following topics lead you through a Serena Release Automation installation.

- Installation Checklist [page 41]
- Installation Recommendations [page 42]
- System Requirements [page 43]
- Database Installation [page 45]
- Server Installation [page 49]
- Single Sign On (SSO) with Serena Business Manager (SBM) [page 60]
- Agent Installation [page 63]
- Installing Agent Relays [page 69]

Installation Checklist

A basic configuration consists of a server, a database, and at least one agent. In production environments, each of these should be installed on separate machines.

The following table summarizes basic installation steps.

Step	Description
1. Review installation recommendations and system requirements	Review the system requirements and installation recommendations.
2. Download Serena Release Automation installation files	Download the server, agent, agent relay, and other installation packages as needed from the Serena Release Automation support website at http://support.serena.com/Case/CaseHome.aspx.
3. Install the database	Create an empty database for Serena Release Automation. Serena Release Automation supports Oracle, MySQL, and Microsoft SQL Server. Complete database installation by configuring the appropriate JDBC driver (typically supplied by the database vendor). **Note:** The installation package includes a lightweight database, Derby, that can be used for evaluation purposes.

Step	Description
4. Install the server	Before you run the installer, you will need to have Java 6 or above installed, the HOME value set, and the empty database created. The installer will prompt you to supply values for the IP address, ports for HTTP communication (secured and unsecured), port for agent communication, and URL.
5. Install agents	Agents are installed on target machines and communicate with the server. When installing an agent, you supply several values defined during server installation. An agent requires various access privileges for the machine where it is installed, which are described in that section.
6. Install agent relays (optional)	If you are using an agent relay, install the relay.
7. Validate the installation	Start the server and agents. To determine if the agent is in communication with the server, display the web application's Resource pane. A value of `Online` in the agent's Status field means the agent is successfully connected. See Chapter 6: Running Serena Release Automation [page 73].
8. Configure your Release Automation system as needed	Configure Server Communication, SSL, Single Sign On (SSO), File Versioning, Users, Groups, Roles and so on as needed for your organization

Installation Recommendations

For optimal performance, Serena recommends the following:

- **Install the server as a user account.** The server should be installed as a dedicated system account whenever possible. While not recommended, Serena Release Automation can run as the root user (or local system user on Windows) and running in this manner avoids all permission errors.
- **Install each agent as a dedicated system account.** Ideally, the account should only be used by Serena Release Automation. Because Serena Release Automation agents are command execution engines, it is advisable to limit what they can do on host machines by creating dedicated users and then granting them appropriate privileges. If you install an agent as the root user (or local system user on Windows), ensure that agent processes cannot adversely affect the host file system.
- **Except for evaluation purposes, do not install an agent on the Serena Release Automation server machine.** Because the agent is resource intensive, installing one on the server machine can degrade performance during large deployments.

- **Install a single agent per host machine.** Multiple agents on the same machine can negatively impact each other's performance. When you must install multiple agents, you might see performance degradation when multiple agents are busy simultaneously.

System Requirements

Serena Release Automation will run on Windows and UNIX-based systems. While the minimum requirements provided below are sufficient for an evaluation, you will want server-class machines for production deployments.

Important:

Ensure that Java is installed on the server machines and on agent relay machines if used. All server and agent relay machines require Java JRE 6 or greater.

Set the JAVA_HOME environment variable to point to the directory you intend to use. A JDK can be used.

This step does not apply to agent machines because the agent installer installs a JRE.

Server Minimum Installation Requirements

- **Windows**: Windows 2000 Server (SP4) or later
- **Processor**: Single core, 1.5 GHz or better
- **Disk Space**: 300 MB or more
- **Memory**: 2 GB, with 256 MB available to Serena Release Automation
- **Java version**: JRE 6 or greater

Server Installation Recommendations

- **Multiple server-class machines**: For production environments, it is recommended to use a multiple-server, active-active approach to distribute the processing load of agent communication.
- **Separate machine for the database**
- **Processor**: 2 CPUs, 2+ cores for each
- **RAM**: 8 GB
- **Storage**: Individual requirements depend on usage, retention policies, and application types. In general, the larger number of artifacts kept in Serena Release Automation's artifact repository (CodeStation), the more storage needed.

  Note: CodeStation is installed when the Serena Release Automation server is installed.

  For production environments, use the following guidelines to determine storage requirements:

- 10-20 GB of database storage should be sufficient for most environments.
- To calculate CodeStation storage requirements:

  ```
  average artifact size * number of versions imported per day * average
  number of days before cleanup
  ```

- Approximately 1MB per deployment of database storage; varies based on local requirements.

  For further assistance in determining storage requirements, contact Serena Customer Support.

- **Network** Gigabit (1000) Ethernet with low-latency to the database.

Agent Minimum Requirements

Designed to be minimally intrusive (typically, an idle agent uses 5Mz CPU), agents require 64-256 MB of memory and 100 MB of disk space. Additional requirements are determined by the processes the agent will run.

For a simple evaluation, the agent can be installed on the same physical machine as the server.

Important: In production environments, Serena *highly* recommends installing agents on separate machines.

Database Requirements

Your relational database is a critical element for performance and disaster recovery. The provided Derby database, while sufficient for proof-of-concept work, is generally insufficient for the enterprise. Full-featured databases like Oracle, MS SQL Server, or MySQL are better options. Ideally, the database—whichever is used—should be configured for high-availability, high-performance, and be backed-up regularly.

10-20 GB of database storage should be sufficient for most environments. For Oracle, an architecture based on Oracle RAC is recommended; for Microsoft SQL Server, a clustered configuration is preferred; for MySQL, use MySQL Cluster.

32- and 64-bit JVM Support

The Serena Release Automation server and agent installers both contain only a 32-bit JVM for all platforms except for Linux 64-bit, which contains only the 64-bit JVM.

Because Serena Release Automation does not require a multi-gigabyte heap, there is little advantage to using a 64-bit JVM. However, there are exceptions. For details, see the Server JVM Support table [page 0] and the Agent JVM Support table [page 0].

Table 1. Server JVM Support table

Operating System	JVM 32-bit	JVM 64-bit
Windows 32-bit	yes	Does not apply
Windows 64-bit*	yes	yes

Operating System	JVM 32-bit	JVM 64-bit
*Windows 2003 64-bit	must use	not supported
UNIX 32-bit	yes	Does not apply
UNIX 64-bit	yes	yes

Table 2. Agent JVM Support table

Operating System	JVM 32-bit	JVM 64-bit
Windows 32-bit	yes	Does not apply
Windows 64-bit	yes	yes
Windows 2003 64-bit	yes	yes
UNIX 32-bit	yes	does not apply
UNIX 64-bit	yes	yes

Database Installation

In addition to the supplied Derby database, Serena Release Automation currently supports Oracle, SQL Server, and MySQL (see Installing Oracle [page 46], Installing MySQL [page 47], or Installing Microsoft SQL Server [page 47]).

Installing Oracle

Before installing the Serena Release Automation server, install an Oracle database. If you are evaluating Serena Release Automation, you can install the database on the same machine where the Serena Release Automation server will be installed.

When you install Serena Release Automation, you will need:

- the Oracle connection information, and
- a user account with table creation privileges.

Serena Release Automation supports the following editions:

- Oracle Database Enterprise
- Oracle Database Standard
- Oracle Database Standard One
- Oracle Database Express

Version 10g or later is supported for each edition.

To install an Oracle database:

1. Download and install the JDK–not a JRE–that corresponds to your operating system (see http://www.oracle.com/technetwork/java/javase/downloads/).
2. Download the Oracle JDBC driver specific to the database edition you are using (see http://www.oracle.com/technetwork/indexes/downloads/index.html) and save in your Apache-Tomcat `libs` folder.
3. Install the JDBC driver.
4. Create the Oracle database by executing the following commands:

   ```
   CREATE USER serena_ra IDENTIFIED by serena_ra;

   GRANT CONNECT TO serena_ra;

   GRANT RESOURCE TO serena_ra;
   ```
5. Begin server installation, see Server Installation [page 49].

   Select the Oracle database option in the installer.
6. Provide the Oracle JDBC driver (see step 2).
7. Provide the JDBC driver class Serena Release Automation will use to connect to the database.

   The default value is:

   ```
   oracle.jdbc.driver.OracleDriver
   ```
8. Provide the JDBC connection string. The format depends on the JDBC driver. Typically, it is similar to:

   ```
   jdbc:oracle:thin:@[DB_URL]:[DB_PORT]/[SID]
   ```

For example:

```
jdbc:oracle:thin:@localhost:1521/dim12
```

9. Finish by entering the database username and password.

Note: The schema name must be the same as the user name.

Installing MySQL

Before installing the Serena Release Automation server, install MySQL. If you are evaluating Serena Release Automation, you can install the database on the same machine where the Serena Release Automation server will be installed.

When you install Serena Release Automation, you will need the MySQL connection information, and a user account with table creation privileges.

To install the MySQL database:

1. Create a database by executing the following commands:

   ```
   CREATE DATABASE serena_ra;

   GRANT ALL ON serena_ra.* TO 'serena_ra'@'%'

   IDENTIFIED BY 'password' WITH GRANT OPTION;
   ```

2. Obtain the MySQL JDBC driver.

   The JDBC jar file is included among the MySQL installation files. The driver is unique to the edition you are using.

3. Begin server installation, see Server Installation [page 49].

   Select the MySQL database option in the installer.

4. Provide the MySQL JDBC driver (see step 2).

   The default value is `com.mysql.Driver`.

5. Provide the JDBC connection string. Typically, it is similar to:

   ```
   jdbc:mysql[DB_URL]:[DB_PORT]:[DB_NAME]
   ```

   For example:

   ```
   jdbc:mysql://localhost:3306/serena_ra
   ```

6. Finish by entering the database username and password.

Installing Microsoft SQL Server

Before installing the Serena Release Automation server, install a SQL Server database. If you are evaluating Serena Release Automation, you can install the database on the same machine where the Serena Release Automation server will be installed.

When you install Serena Release Automation, you will need the SQL Server connection information, and a user account with table creation privileges.

To install the MS SQL Server database:

1. Download and install the JDK–not a JRE–that corresponds to your operating system (see http://www.oracle.com/technetwork/java/javase/downloads/).
2. Download the SQL Server JDBC driver from http://www.microsoft.com/en-gb/download/ and save it in your Apache-Tomcat `libs` folder.
3. Install the JDBC driver.
4. Using the SQL Server Management Studio, execute the following commands:

   ```
   CREATE DATABASE serena_ra;

   USE serena_ra;

   CREATE LOGIN serena_ra WITH PASSWORD = 'password';

   CREATE USER serena_ra FOR LOGIN serena_ra WITH DEFAULT_SCHEMA =
   serena_ra;

   CREATE SCHEMA serena_ra AUTHORIZATION serena_ra;

   GRANT ALL TO serena_ra;
   ```

5. Begin server installation, see Server Installation [page 49].
6. Select the SQL Server database option in the installer.
7. Provide your SQL Server database details:

Table 1. MS SQL Server Database Installation Details table

Field	Description
JDBC driver jar filename	Browse to and select the JDBC driver jar file you downloaded in step 2.
JDBC driver class	The default value is: `com.microsoft.sqlserver.jdbc.SQLServerDriver`
Database connection string	The format depends on the JDBC driver. Typically, it is similar to: `jdbc:sqlserver://[DB_URL]:[DB_PORT];databaseName=[DB_NAME]` For example, `jdbc:sqlserver://localhost:1433;databaseName=serena_ra`
User	Database username. The default is `serena_ra`.
Password and **Confirm password**	Database username password, and re-entry for confirmation.

Server Installation

The server provides services such as the user interface used to configure application deployments, the work flow engine, the security service, and the artifact repository, among others.

Important: If you are installing the server in a production environment, install and configure the database you intend to use *before* installing the server. See Database Installation [page 45].

Downloading Serena Release Automation

The installation package is available from the Serena support portal. If you need help accessing the portal, contact your Serena Sales representative.

Note: You must have a license in order to download the product. For an evaluation license, please contact your Serena Sales Representative.

1. Go to http://support.serena.com/Case/CaseHome.aspx and log in using your customer account.

   If you do not have an account, please contact your Sales Representative.

2. Browse to the **My Downloads** tab.

3. From the **Please Select Product** drop-down, select **Serena Release Automation**.

4. Select the Serena Release Automation version you want to download.

5. Select the appropriate package for your environment for the server, agent, and agent relay.

   Serena Release Automation enables you to install agents on any supported platform, regardless of the operating system where the server is installed.

A license file should have been provided to you by your Serena Sales Representative.

Interactive Server Installation (Windows, Linux/UNIX (AIX, Solaris)

This type of installation uses a wizard that guides you through the complete process. The properties set during the server installation are recorded in the `installed.properties` file located in the `server_install/conf/server/` directory.

Note:

- (Unix/Linux) Root privileges are required to install the Serena Release Automation server; the installer writes product registration data into the `/var/opt/` directory.

To install the server:

Download and run the installer `SerenaRA-Server.exe`, or your platform-specific alternative, and follow the step-by-step instructions. See also:

- (Windows) Server Install: Destination Folder Panel [page 50]

- (Windows) Server Install: Administrator Details Panel [page 51]

(Windows) Server Install: Destination Folder Panel

The following table describes the fields that appear on the Windows server install wizard, Destination Folder panel:

Field	Default	Description
Install Serena Release Automation to:	C:\Users\<username>\.serena\ra	Directory to install the Serena Release Automation server. **Note:** You cannot use redirected drives for this path because Serena Common Tomcat starts as a Windows service. See the Serena Support Knowledgebase Solution S139852 if this is a concern for you.
Use existing settings	not selected	For upgrading or reinstall. If you want to install the server using the existing settings and maintain the database, select this option.
Skip database creation	not selected	For upgrading or reinstall. If you are using any database other than Derby and you want to install the server using the existing settings and maintain the existing database, select this option. **Note:** You will need to provide connectivity information to get access to an existing database.
JMS Connections port	7918	Specifies the port on which the Serena Release Automation agents you install will make JMS Connections to the Serena Release Automation server.

Field	Default	Description
Agent Mutual Authentication	not selected	If selected, agent will use SSL mutual authentication. If not selected, agent will use unauthenticated mode; communication is encrypted but users do not have to authenticate or verify their credentials. For more information, see Chapter 8: SSL Configuration [page 81].

(Windows) Server Install: Administrator Details Panel

The following table describes the fields that appear on the Windows server install wizard, Administrator Details panel:

Field	Default	Description
Administrator User	admin	Specify the installation owner's administration user name. This will be the Username required to log into the Server the first time.
Password	does not apply	(Required) User-defined password for the Administrator User name entered in the previous field. This will be the password needed to log into the Server the first time.
Confirm Password	does not apply	(Required) Re-enter the password exactly as you entered it in the Password field.

Server Installation (Other UNIX Platforms)

These server installation instructions are for UNIX platforms other than Linux/UNIX (AIX, Solaris).

To install the server:

1. Download the `serena_ra.war` file. This file is contained in the `Serena.RA-Other40.zip`. For more information, see Downloading Serena Release Automation [page 49].

2. Stop Apache Tomcat.

3. Copy and paste the `serena_ra.war` file into your Tomcat 1.6 or higher or Serena Common Tomcat `webapps` folder.

4. Restart Tomcat.

5. Launch a browser and navigate to http://localhost:8080/serena_ra.

   The configuration setup displays.

6. Modify any configuration parameters as needed:
   - Install location
   - External Web URL
   - Agent port and choice of mutual authentication
   - Database (port, schema, user, password)
7. Click Install.

When the server installation is complete, the Serena Release Automation login page displays.

Silent Mode Server Installation

This section contains information about how to implement a silent install of the Serena Release Automation server on Windows (see (Windows) Server Silent Installation [page 52]),and Linux/UNIX (AIX, Solaris), see (Linux/UNIX Platforms) Server Silent Install [page 56].

(Windows) Server Silent Installation

To install the server in silent mode:

1. Download the installation files. For more information, see Downloading Serena Release Automation [page 49].
2. Create an options file and save as `C:/optiontsFile.txt` (see (Windows) Server Silent Install Options [page 52]).
3. In Windows, issue the command:

   ```
   cmd /c SerenaRA-Server.exe /s /V"/qn /L*vx "%TEMP%\silent-install.log"
   PROPFILE=\"c:\optionsFile.txt\" "
   ```

   Where:

   - `"%TEMP%\silent-install.log"` is an absolute path to a logfile.
   - `PROPFILE=\"c:\optionsFile.txt\"` is an absolute path to the options file you created in Step 2.
   - You have included the **required** space between the final two quotes.

For examples of the `optionsFile.txt`, see (Windows) Server Silent Install: OptionsFile.txt Examples [page 54]).

(Windows) Server Silent Install Options

Option	Default/if not specified	Description
`AgreeToLicense`	`No`	must be set to `Yes`

Option	Default/if not specified	Description
USE_EXISTING_SETTINGS		Set to UseExisting if you want to skip the database install and administration part of the installation. **Important:** Use only if there are existing Serena Release Automation settings populated in the directory specified by: SRA_USER_INSTALLDIR.
SRA_USER_INSTALLDIR	c:\Documents and Settings\→ Administrator\→ .serena\ra	Directory to install the Serena Release Automation server, or if using the USE_EXISTING_SETTINGS=→ "UseExisting" option, the directory where your Serena Release Automation settings already exist.
SKIP_DB		To skip database creation, include and set this option: SKIP_DB=SkipDB **Note:** To use this option, the SRA_USER_INSTALLDIR must *not* already exist.
AGENT_MUTUAL_AUTH	N	agent mutual authentication option: y|N For more information, see Chapter 8: SSL Configuration [page 81].
JMS_PORT	7918	server port
DB_TYPE	derby	Use to specify a database vendor other than the default: MYSQL|ORA|SQLSVR
<DB>_USER	serena_ra	database user ID, where <DB> is: DERBY|MYSQL|ORA|SQLSVR
<DB>_PASSWORD	password	password for database user ID, where <DB> is: DERBY|MYSQL|ORA|SQLSVR
ORA_DB_SCHEMA		(Required for Oracle) database schema
<DB>_JDBC_DRIVER		(Required)For database other than Derby, JDBC database driver file, where <DB> is DERBY|MYSQL|ORA|SQLSVR

Option	Default/if not specified	Description
DERBY_PORT		(Required for Derby) specify the Derby port 11377
<DB>_DB_CONN		(Required)For database other than Derby, database connection, where <DB> is MYSQL|ORA|SQLSVR
IS_INSTALL_MODE		(Required) must be set to: "silent"
SRA_ADMIN		(Required) specify the installation owner's administration user name.
SRA_ADMIN_PWD		(Required) user-defined password for the user name set by the SRA_ADMIN option.
TC_PORT	8080	Tomcat port. Only used for new Serena Common Tomcat installations.

For examples of the option settings needed for each database (Derby, Oracle, MySQL, and MS SQL Server), see (Windows) Server Silent Install: OptionsFile.txt Examples [page 54].

(Windows) Server Silent Install: OptionsFile.txt Examples

This section contains examples of how to configure the install options in an optionsFile.txt file for a server silent install on Windows.

Derby Database optionsFile.txt example

```
AgreeToLicense=Yes
USE_EXISTING_SETTINGS=""
SRA_USER_INSTALLDIR=C:\Documents and Settings\Administrator\.serena\ra
SKIP_DB=""
AGENT_MUTUAL_AUTH=""
JMS_PORT=7918
DERBY_PORT=11377
DERBY_USER=serena_ra
DERBY_PASSWORD=serena_ra
IS_INSTALL_MODE=silent
SRA_ADMIN=admin1234
SRA_ADMIN_PWD=password123
TC_PORT=8088
```

Note: All defaults are used except for the Tomcat Port which is set to 8088.

"Use Existing Settings" optionsFile.txt example

```
AgreeToLicense=Yes
USE_EXISTING_SETTINGS=UseExisting
SRA_USER_INSTALLDIR=C:\Documents and Settings\Administrator\.serena\ra
SRA_ADMIN=admin1234
SRA_ADMIN_PWD=password123
TC_PORT=8080
```

Note: A Serena Release Automation server installation location already exists as specified by the SRA_USER_INSTALLDIR option. Using this optionsFile.txt, the server will be installed and the existing settings will be used for the install.

MySQL Database optionsFile.txt example

```
AgreeToLicense=Yes
SRA_USER_INSTALLDIR=C:\Documents and Settings\Administrator\.serena\ra
SKIP_DB=""
AGENT_MUTUAL_AUTH=""
JMS_PORT=7918
DB_TYPE=mysql
JDBC_DRIVER_SOURCE=c:\TestArea\libloc\mysql-connector-java-5.1.24-bin.jar
MYSQL_JDBC_DRIVER=com.mysql.jdbc.Driver
MYSQL_DB_CONN=jdbc:mysql://localhost:3306/serena_ra
MYSQL_USER=serena_ra
MYSQL_PASSWORD=passwordabc
IS_INSTALL_MODE=silent
SRA_ADMIN=admin1234
SRA_ADMIN_PWD=password123
TC_PORT=8080
```

Oracle Database optionsFile.txt example

```
...
...
DB_TYPE=oracle
JDBC_DRIVER_SOURCE=c:\TestArea\libloc\oracle-connector-ojdbc5.jar
ORA_JDBC_DRIVER=oracle.jdbc.driver.OracleDriver
ORA_DB_CONN=jdbc:oracle:thin:@localhost:1521/serena_ra
ORA_DB_SCHEMA=serena_ra
ORA_PASSWORD=password123...
```

MS SQL Server Database optionsFile.txt example

```
...
...
DB_TYPE=sqlserver
SQLSVR_JDBC_DRIVER=com.microsoft.sqlserver.jdbc.SQLServerDriver
SQLSVR_DB_CONN=jdbc:sqlserver://localhost:1433;DatabaseName=serena_ra
SQLSVR_USER=serena_ra
SQLSVR_PASSWORD=mypassword...
```

"Skip Database Creation" optionsFile.txt example

```
AgreeToLicense=Yes
SRA_USER_INSTALLDIR=C:\Documents and Settings\Administrator\.serena\ra
SKIP_DB=SkipDB
AGENT_MUTUAL_AUTH=""
JMS_PORT=7918
DB_TYPE=sqlserver
SQLSVR_JDBC_DRIVER=com.microsoft.sqlserver.jdbc.SQLServerDriver
SQLSVR_DB_CONN=jdbc:sqlserver://localhost:1433;DatabaseName=serena_ra
SQLSVR_USER=serena_ra
SQLSVR_PASSWORD=mypassword
IS_INSTALL_MODE=silent
TC_PORT=8080
```

Note: The SRA_USER_INSTALLDIR specified must *not* already exist. Using this optionsFile.txt because the SKIP_DB=skipDB option is specified, the DB_TYPE option cannot be set to `derby`, and the SRA_ADMIN and SRA_ADMIN_PWD options cannot be set during the server installation.

(Linux/UNIX Platforms) Server Silent Install

To install the server in silent mode:

1. Download the installation files. For more information, see Downloading Serena Release Automation [page 49].

2. Create an `optionsFile.txt` and save it to the root directory (see (Linux/UNIX) Server Silent Install Options.

3. In Linux/UNIX(AIX, Solaris), as a user with root privileges issue the command:

   ```
   SerenaRA-server.bin -silent -options optionsFile.txt
   ```

   Where:

   `optionsFile.txt` is a file that contains the properties you have set for your system.

For examples of the `optionsFile.txt`, see (Linux/UNIX) Server Silent Install: optionsFile.txt Examples [page 59].

(Linux/UNIX) Server Silent Install Options

This section provides the list and description of the Linux/UNIX (AIX, Solaris) silent install options, and examples of the option settings needed for each database (Derby, Oracle, MySQL, and MS SQL Server).

For examples of the `optionsFile.txt`, see (Linux/UNIX) Server Silent Install: optionsFile.txt Examples [page 59].

(Linux/UNIX) Server silent install options table

Option	Default	Description
`-V IS_SELECTED_INSTALLATION_TYPE`	`typical`	must be set to `typical` (do not enclose in quotes)
`-P installLocation`		Directory to install the Serena Release Automation server. Should be set to `"/opt/serena/sra"`
`-V IS_DESTINATION`		Directory to install the Serena Release Automation server. Should be set to `"/opt/serena/sra"`
`-V ServerDetailsUseExisting`	`false`	To skip the database install and administration part of the installation, include and set to `"true"`. **Important:** Use only if there are existing Serena Release Automation settings populated in the directory specified by: `"SRA_USER_INSTALLDIR"`.
`-V ServerDetailsSkipDb`	`false`	To skip database creation, include and set this option to `"true"`.
`-V ServerDetailsPort`	`7918`	server port
`-V ServerDetailsMutualAuth`	`false`	agent mutual authentication option: **`"false"`**`|true` For more information, see Chapter 8: SSL Configuration [page 81].
`-V SctFoundLoc`		To reuse an existing Serena Common Tomcat/Tools, specify and set to `true`. When set to `true`, you *must* also include the `"SctInstallLoc"` option.

Option	Default	Description
`-V SctInstallLoc`		(Required) If you specify the "`SctFoundLoc`" option to reuse an existing Serena Common Tomcat/ Tools, you must use this option to specify the location of the existing Serena Common Tomcat/Tools.
`-V DbDetailsVendor`	`derby`	Use to specify a database vendor other than the default: `oracle|mysql|sqlserver`
`-V DbDetailsUser`	`serena_ra`	database user ID
`-V DbDetailsPwd`	`password`	password for database user ID
`-V DbDetailsSchema`		(Required for Oracle) database schema
`-V DbDetailsDriver`		database driver class, for example: `org.apache.derby.jdbc.→ ClientDriver`
`dbDetailsDriverFilename`		Required for a database other than Derby; specify a database driver file.
`-V DbDetailsDerbyPort`		(Required for Derby) specifies the Derby port `11377`
`-V DbDetailsConnection`		database connection
`-V IS_INSTALL_MODE`		must be set to: "`silent`"
`-V AdminDetailsName`		Specifies the installation owner's administration user name.

Option	Default	Description
`-V AdminDetailsPwd`		(Required) user-defined password for the username set by the `AdminDetailsName` option.
`-V SctTomcatOwner`		system user to own Serena Common Tomcat files; valid for new Serena common Tomcat scenario.
`-V SctTomcatPort`	`8080`	Tomcat port. Only used for new Serena Common Tomcat installations. See the SctFoundLoc option.

(Linux/UNIX) Server Silent Install: optionsFile.txt Examples

This section contains examples of how to configure the install options in an `optionsFile.txt` file for a server silent install on Linux/UNIX (AIX, Solaris).

Derby Database `optionsFile.txt` example

Note: This options file reuses an existing Serena Common Tomcat installation (`SctFoundLoc="true"`). The location of the existing Tomcat is specified by `SctInstallLoc="/opt/serena/common"`.

```
-V IS_SELECTED_INSTALLATION_TYPE=typical
-P installLocation="/opt/serena/sra"
-V IS_DESTINATION="/opt/serena/sra"
-V SctFoundLoc="true"
-V SctInstallLoc="/opt/serena/common"
-V DbDetailsVendor="derby"
-V DbDetailsUser="User01"
-V DbDetailsPwd="MyPassword"
-V DbDetailsDriver="org.apache.derby.jdbc.ClientDriver"
-V DbDetailsDerbyPort="11377"
-V DbDetailsConnection="jdbc\:derby\://localhost\:11377/data"
-V ServerDetailsMutualAuth="false"
-V IS_INSTALL_MODE="silent"
-V AdminDetailsName="admin123"
-V AdminDetailsPwd="mypassword123"
```

MySQL Database `optionsFile.txt` example

```
-V IS_SELECTED_INSTALLATION_TYPE=typical
-P installLocation="/opt/serena/sra"
-V IS_DESTINATION="/opt/serena/sra"
-V DbDetailsVendor="mysql"
-V DbDetailsUser="sra"
-V DbDetailsPwd="sra"
```

```
-V DbDetailsDriver="com.mysql.jdbc.Driver"
-V DbDetailsConnection="jdbc\:mysql\://localhost\:3306/serena_ra"
-V ServerDetailsMutualAuth="false"
-V IS_INSTALL_MODE="silent"
-V AdminDetailsName="admin123"
-V AdminDetailsPwd="mypassword123"
-V SctTomcatOwner=dmsys
-V SctTomcatPort="8080"
```

Oracle Database `optionsFile.txt` example

```
-V IS_SELECTED_INSTALLATION_TYPE=typical
-P installLocation="/opt/serena/sra"
-V IS_DESTINATION="/opt/serena/sra"
-V DbDetailsVendor="oracle"
-V DbDetailsUser="User02"
-V DbDetailsPwd="MyPassword"
-V DbDetailsSchema="serena_ra"
-V DbDetailsDriver="oracle.jdbc.driver.OracleDriver"
-V DetailsConnection="jdbc\:oracle\:thin\:@localhost\:1521"
-V ServerDetailsMutualAuth="false"
-V IS_INSTALL_MODE="silent"
-V AdminDetailsName="admin123"
-V AdminDetailsPwd="mypassword123"
-V SctTomcatOwner="dmsys"
-V SctTomcatPort="8080"
```

MS SQL Server Database `optionsFile.txt` example

```
-V IS_SELECTED_INSTALLATION_TYPE=typical
-P installLocation="/opt/serena/sra"
-V IS_DESTINATION="/opt/serena/sra"
-V DbDetailsVendor="sqlserver"
-V DbDetailsUser="User03"
-V DbDetailsPwd="MyPassword"
-V DbDetailsDriver="com.microsofl.sqlserver.jdbc.SQLServerDriver"
-V DbDetailsConnection="jdbc\:sqlserver\://localhost\:1433;DatabaseName=serena_ra"
-V ServerDetailsMutualAuth="false"
-V IS_INSTALL_MODE="silent"
-V AdminDetailsName="admin123"
-V AdminDetailsPwd="mypassword123"
-V SctTomcatOwner="dmsys"
-V SctTomcatPort="8080"
```

Single Sign On (SSO) with Serena Business Manager (SBM)

Single Sign On (SSO) refers to Serena-installed software that enables a user to log in to a Web-based component of SBM and be recognized on subsequent accesses to that component or other Web-based components of SBM. This software also provides the ability for security tokens to be used in an orchestration, allowing Web services to be called without requiring the user to provide credentials at inconvenient times. For more information, see Integrating with Serena Business Manager using SSO [page 61].

Integrating with Serena Business Manager using SSO

To configure Serena Release Automation for Single Sign On (SSO), you must complete the following:

1. Install and configure Serena Business Manager (SBM), see *SBM Installation and Configuration Guide*.
2. Install Serena Release Automation with Serena Common Tomcat, see Chapter 5: Installing Servers and Agents [page 41].
3. Configure Tomcat for SSO, see Configuring Tomcat for SSO [page 61].

Configuring Tomcat for SSO

You must have the Serena Release Automation server installed on the same machine as the Serena Common Tomcat.

1. On the Serena Release Automation server, stop the Serena Common Tomcat service.
2. Navigate to the `..\Serena\..\Common Tools\tomcat\6.0\alfssogatekeeper\conf` directory.
3. In `gatekeeper-core-config.xml`, change the following parameters if necessary to replace the host and port values, shown here as `[host]` and `[port]`, with the host and port for your SBM server.

   ```
   <parameter name="SecurityTokenService"
   Type="xsd:anyURI">HTTP://[host]:[port]/TokenService/services/
   Trust<parameter>

   <parameter name="SecurityTokenServiceExternal"
   Type="xsd:anyURI">HTTP://[host]:[port]/TokenService/services/
   Trust</parameter>

   <parameter name="FederationServerURL"
   Type="xsd:anyURI">HTTP://[host]:[port]/ALFSSOLogin/login</parameter>
   ```

4. Modify the `gatekeeper-services-config.xml` file as follows:

   ```
     <GatekeeperProtectionControl>
        <ProtectedURIs>
          <URIMatcher requestURI="/dimensions/*"/>
    ...
       <URIMatcher requestURI="/serena_ra/*"/>
    ....
        </ProtectedURIs>
    ...
      </GatekeeperProtectionControl>
   ```

```
  <ServiceEntryPoints>
    <BrowserRequests>
...
      <URIMatcher requestURI="/serena_ra/*"/>
...
    </BrowserRequests>
...
  </ServiceEntryPoints>
```

5. Navigate to the following directory:
   `..\Serena\..\common\tomcat\6.0\webapps\serena_ra\WEB-INF\`

6. In the `web.xml` file, enable the SSO filters by uncommenting the Serena SSO Gatekeeper Filter Configuration section:

```
- <filter>
  <filter-name>ALFSSOGatekeeperFilter</filter-name>
  <filter-class>org.eclipse.alf.security.sso.server.gatekeeper.filterloader.
GatekeeperFilterLoader</filter-class>
- <init-param>
  <param-name>gatekeeper.enabled</param-name>
  <param-value>true</param-value>
  </init-param>
- <init-param>
  <param-name>gatekeeper.config.filename</param-name>
  <param-value>${catalina.home}/alfssogatekeeper/conf/
gatekeeper-services-config.xml</param-value>
  </init-param>
- <init-param>
  <param-name>gatekeeper.lib.dir</param-name>
  <param-value>${catalina.home}/alfssogatekeeper/lib</param-value>
  </init-param>
- <init-param>
  <param-name>gatekeeper.root.dir</param-name>
  <param-value>${catalina.home}/alfssogatekeeper</param-value>
  </init-param>
- <init-param>
  <param-name>gatekeeper.log4j.use-repo-selector</param-name>
  <param-value>true</param-value>
  </init-param>
- <init-param>
  <param-name>gatekeeper.log4j.create-new-repo</param-name>
  <param-value>true</param-value>
  </init-param>
- <init-param>
  <param-name>gatekeeper.log4j.properties.filename</param-name>
  <param-value>${catalina.home}/alfssogatekeeper/conf/log4j.properties
</param-value>
  </init-param>
  </filter>
- <filter-mapping>
  <filter-name>ALFSSOGatekeeperFilter</filter-name>
  <url-pattern>/*</url-pattern>
```

```
        </filter-mapping>
```

7. On the Serena Release Automation server, start the Serena Common Tomcat service.

8. Verify using the Serena Release Automation Web client URL: `http://<host>:<CT_port>/serena_ra/`, where *`CT_port`* is the Common Tools http port.

   Instead of the default Serena Release Automation login page, the SBM Single Sign-On page should display.

   Enter your Username and Password to access Serena Release Automation.

   See also, Single Sign Out [page 63].

Single Sign Out

When you use Single Sign On (SSO), single sign out will work correctly as long as you have the Serena Release Automation server and the SSO server both configured to use the same host. For more information, see Single Sign On (SSO) with Serena Business Manager (SBM) [page 60].

Agent Installation

For production environments, Serena recommends creating a user account dedicated to running the agent on the machine where the agent is installed.

For simple evaluations, the administrative user can run the agent on the machine where the server is located. But if you plan to run deployments on several machines, a separate agent should be installed on each machine. If, for example, your testing environment consists of three machines, install an agent on each one. Follow the same procedure for each environment the application uses.

Important: Except for evaluation purposes, do *not* install an agent on the same machine as the server.

Each agent needs the appropriate rights to communicate with the Serena Release Automation server.

At a minimum, each agent should have permission to:

- **Create a cache.** By default, the cache is located in the home directory of the user running the agent. The cache can be moved or disabled.

- **Open a TCP connection**. The agent uses a TCP connection to communicate with the server's JMS port.

- **Open a HTTP(S) connection.** The agent must be able to connect to the Serena Release Automation user interface in order to download artifacts from the CodeStation repository.

- **Access the file system**. Many agents need read/write permissions to items on the file system.

Interactive Agent Installation (Windows, Linux/UNIX (AIX, Solaris, HP-UX)

This type of installation uses a wizard that guides you through the complete process.

To install the agent:

1. Download the installation files. For more information, see Downloading Serena Release Automation [page 49].

2. Run the downloaded installer `SerenaRA-agent.exe`, or your platform-specific alternative, and follow the step-by-step instructions.

   Note: Root privileges are required to install the Serena Release Automation server on a UNIX server; the installer writes product registration data into the `/var/opt/` directory.

Silent Mode Agent Installation

This section contains information about how to implement a silent install of the Serena Release Automation agent on Windows, Linux/UNIX (AIX, Solaris, HP-UX).

Important: Before you can implement a silent install of an agent, you *must* create and save an `optionsFile.txt`.

(Windows) Agent Silent Installation

To install an agent in silent mode:

1. Download the appropriate agent installer for your platform. For more information, see Downloading Serena Release Automation [page 49].

2. Create an options file and save as:

   ```
   C:/optiontsFile.txt
   ```

   Note: Your optionsFile should be in ANSI format; the agent installer does not support UTF.

3. Issue the command:

   ```
   SerenaRA-agent.exe -silent -options optionsFile.txt
   ```

   where:

   ```
   optionsFile.txt
   ```

   is the options file you created in Step 2.

(Windows) Agent Silent Install Options

Option	Description
`-P installLocation`	(Required) Location into which the agent will be installed.

Option	Description
`-V IS_DESTINATION`	(Required) Location into which the agent will be installed.
`-V AgentName`	(Required) Name of the agent.
`-V AgentOwner`	(Linux/UNIX Required) User name of agent installation owner.
`-V UseRelayYes`	(Required) To use an agent relay set to `true`, if not, set to `false`.
`-V RelayHost`	(Required only if `-V UseRelayYes=true` is specified.) The hostname of the agent relay to be used.
`-V RelayPort`	(Required only if `-V UseRelayYes=true` is specified.) The port number of the agent relay to be used.
`-V RelayProxyPort`	(Required only if `-V UseRelayYes=true` is specified.) The proxy port number of the agent relay.
`-V ServerHost`	(Required if `-V UseRelayYes=false`) The hostame of the release automation server.
`-V ServerPort`	(Required if `-V UseRelayYes=false`) The port number of the release automation server.
`-V ServerMutAuth`	(Required) If mutual authentication will be used with the server set to `true`, otherwise set to `false`.
`-V JreNew`	(Required) To install a new JRE with the agent, set to `true`, if not then set to `false`.
`-V JreInstallLoc`	(Required if `-V JreNew=false`) To specify the location of a pre-existing JRE for the agent to use.

Option	Description
`-V ServiceYes`	(Windows Required) To create a Windows service for the agent, set to `true`, otherwise set to `false`. This option is only valid for Windows systems.
Note: The following fields are only required when you specify `-V ServiceYes=True` and only apply to Windows systems.	
`-V ServiceName`	(Required only when `-V ServiceYes=True`) is specified. The name to use for the Windows service.
`-V ServiceStartAuto`	(Required only when `-V ServiceYes=True`) is specified. To configure the Windows service to start automatically, set to `true`, if not set to `false`.
`-V ServiceAccName`	(Required only when `-V ServiceYes=True`) is specified. To specify the log on user account name for the Windows service.
`-V ServiceAccPass`	(Required only when `-V ServiceYes=true`) is specified. To specify the log on user account password for the Windows service.

For examples of the option settings needed for each database (Derby, Oracle, MySQL, and MS SQL Server), see (Windows) Agent Silent Install: OptionsFile.txt Example [page 66].

(Windows) Agent Silent Install: OptionsFile.txt Example

This section contains an example of how to configure the install options in an `optionsFile.txt` file for an agent silent install on Windows.

Example: Agent Windows optionsFile.txt

Using this optionsFile.txt, the agent will be named `agent123`, be configured to connect directly to the server on `serverabc` (not through an agent relay), and will point to a new installation of a JRE. Mutual Authentication mode will be activated for the agent, and the agent will run as a Windows service under the name `SRA-Agent123` that will start automatically when the system starts. The agent Windows service admin user account name will be set to: `admin01`, with the password: `password`.

```
-P installLocation="C:\Program Files (x86)\Serena\Release Automation Agent"
-V IS_DESTINATION="C:\Program Files (x86)\Serena\Release Automation Agent"
-V AgentName="agent123"
-V UseRelayYes="false"
-V ServerHost="serverabc"
-V ServerPort="7918"
-V ServerMutAuth="true"
```

```
-V ServiceYes="true"
-V ServiceName="sra-agent123"
-V ServiceStartAuto="true"
-V ServiceAccName="admin01"
-V ServiceAccPass="password"
-V JreNew="true"
```

(Linux/UNIX Platforms) Agent Silent Installation

To install an agent in silent mode:

1. Download the appropriate agent installer for your platform. For more information, see Downloading Serena Release Automation [page 49].

2. Create an options file and save as:

   `optiontsFile.txt`

   Note: Your optionsFile should be in ANSI format; the agent installer does not support UTF.

3. Issue the command:

   `SerenaRA-agent.exe -silent -options optionsFile.txt`

   where:

   `optionsFile.txt`

   is the options file you created in Step 2.

(Linux/UNIX) Agent Silent Install Options

Option	Description
`-P installLocation`	(Required) Location into which the agent will be installed.
`-V IS_DESTINATION`	(Required) Location into which the agent will be installed.
`-V AgentName`	(Required) Name of the agent.
`-V AgentOwner`	(Linux/UNIX Required) User name of agent installation owner.

Option	Description
`-V UseRelayYes`	(Required) To use an agent relay set to `true`, if not, set to `false`.
`-V RelayHost`	(Required only if `-V UseRelayYes="true"` is specified.) The hostname of the agent relay to be used.
`-V RelayPort`	(Required only if `-V UseRelayYes="true"` is specified.) The port number of the agent relay to be used.
`-V RelayProxyPort`	(Required only if `-V UseRelayYes="true"` is specified.) The proxy port number of the agent relay.
`-V ServerHost`	(Required if `-V RelayPortYes="false"`) The hostame of the release automation server.
`-V ServerPort`	(Required if `-V UseRelayYes="false"`) The port number of the release automation server.
`-V ServerMutAuth`	(Required) If mutual authentication will be used with the server set to `true`, otherwise set to `false`.
`-V JreNew`	(Required) To install a new JRE with the agent, set to `true`, if not then set to `false`.
`-V JreInstallLoc`	(Required if `-V JreNew="false"`) To specify the location of a pre-existing JRE for the agent to use.
`-V ServiceYes`	(Windows Required) To create a Windows service for the agent, set to `true`, otherwise set to `false`. This option is only valid for Windows systems.

For related information, see (Linux/UNIX) Agent Silent Install: OptionsFile.txt Examples [page 68], and (Linux/UNIX Platforms) Agent Silent Installation [page 67].

(Linux/UNIX) Agent Silent Install: OptionsFile.txt Examples

This section contain examples of `optionsFile.txt` file configurations for an agent silent install on a Linux or a UNIX platform.

Example 1: Agent Linux/UNIX platforms optionsFile.txt

Using this optionsFile.txt, during install the agent will be configured to connect directly with the server on `serverabc` (not through an agent relay), will turn mutual authentication mode on for the agent and install a new JRE.

```
-P installLocation="/opt/serena/Release_Automation_Agent"
-V IS_DESTINATION="/opt/serena/Release_Automation_Agent"
-V AgentName="agent123"
-V AgentOwner="Admin01"
-V UseRelayYes="false"
-V ServerHost=serverabc
-V ServerPort="7918"
-V ServerMutAuth="true"
-V JreNew="true"
```

Example 2: Agent Linux/UNIX platforms optionsFile.txt

Using this optionsFile.txt, during installl the agent will be configured to use an agent relay named `relayagent01` on port `7916` to connect to the server. Mutual Authentication for the agent will be turned on, and the agent will be configured to point to and use the JRE that currently exists in: `/opt/Java/`.

```
-P installLocation="/opt/serena/Release_Automation_Agent"
-V IS_DESTINATION="/opt/serena/Release_Automation_Agent"
-V AgentName="agentABC"
-V AgentOwner="Admin02"
-V UseRelayYes="true"
-V RelayHost="relayagent01"
-V RelayPort="7916"
-V ServerMutAuth="true"
-V JreNew="false"
-V JreInstallLoc="/opt/Java/"
```

Installing Agent Relays

An agent relay is a communication proxy for agents that are located behind a firewall or in another network location. As long as there is at least a low bandwidth WAN connection between the server and remote agents, the Serena Release Automation server can send work to agents located in other geographic locations via the relay.

An agent relay requires that only a single machine in the remote network contact the server. Other remote agents communicate with the server by using the agent relay. All agent-server communication from the remote network goes through the relay.

You can download the agent relay installation package from the Serena support portal http://support.serena.com/Case/CaseHome.aspx. Before installing, ensure that:

- Java 1.6.0 or later is installed.
- The server with which the relay will connect is already installed.
- The user account and password created during server installation is available.

To install an agent relay:

1. Expand the compressed installation file.
2. From within the expanded `agent-relay-install` directory run the `install.cmd` script.
3. The installation program will prompt you for the following information. Any default values suggested by the program (displayed within brackets) can be accepted by simply pressing **Enter**. If two options are given, such as `Y/n`, the capitalized option is the default value.
   Table 1. Agent Relay Configuration table

Parameter	Description
Directory where you would like to install the agent relay	Enter the directory where you want the agent relay installed.
Java home	Directory where Java is installed. Ensure that the JAVA_HOME environment variable points to this directory.
Name of this relay	Enter the name of the agent relay. Each relay must have a unique name. The default name is `agent-relay`.
IP or hostname which this agent relay should use	Enter the IP or hostname on which the relay will listen.
Port which this agent relay should proxy HTTP requests on	Enter the port on which the agent relay should listen for HTTP requests coming from agents. The default value is 20080.
Port which this agent relay should use for communication.	Enter the port on which the agent relay will use for JMS-based communications with remote agents. The default value is 7916.
Connect the agent relay to a central server?	Specify whether you want the relay to connect to the Serena Release Automation server.
IP or hostname of your central server	If you indicated that you want to connect the relay to the server, enter the IP or host name where the relay can contact the server.

Parameter	Description
Port which the central server uses for communication	If you indicated that you want to connect the relay to the server, enter the port the server uses to communicate with agents. The default value is 7918.
Use secure communication between the agent, relay and server?	Specify whether you want to use SSL security for communications between server, relay, and remote agents. The default value is `Y`. **Important:** To use the relay, you must answer yes. Answering yes activates SSL security for HTTP- and JMS-based communications. If you answer no, the relay *will not* be able to communicate with the server (which uses JMS for most communications).
Use mutual authentication between the agent, relay and server.	If mutual authentication is required, enter `Y`. See Chapter 8: SSL Configuration [page 81] for information about activating mutual authentication.
Install the Agent Relay as Windows service?	If you are installing the relay on Windows, you can install it as a Windows service. The default value is `N`. If you specify `Y`, see the following Agent Relay Windows Service Configuration table [page 0].

If you choose to install the agent relay as a Windows service, the installation program will also prompt you for the following information.

Note: If you need to modify the relay, you can edit these properties in the `agentrelay.properties` file located in the `relay_installation\conf` directory.

Table 2. Agent Relay Windows Service Configuration table

Parameter	Description
`Windows service name`	(Required) Unique name for the service. Default is `agentrelay`
`User account name for the Windows service`	(Required) An account name, including the domain path to run the service as. For a local user, insert "`.\`" before the user name. Default is `.\localsystem`
`User account password for the Windows service`	Password for the Windows service user account name.

Parameter	**Description**
`Start the service automatically`	(Required) Specify whether or not (`y|N`) you want the Windows service to be started automatically. Default is `N`

Chapter 6: Running Serena Release Automation

The following topics describe how you start the Serena Release Automation services to validate your installation.

- Stopping and Starting the Server [page 73]
- Starting and Stopping an Agent Relay [page 73]
- Starting and Stopping an Agent [page 74]
- Accessing Serena Release Automation [page 74]

Stopping and Starting the Server

The installer installs Serena Release Automation under the Serena Common Tomcat Web server and starts the service automatically.

To stop or start the server service:

- In Windows, use `Administrative Tools > Services` to start or stop **Serena Common Tomcat**.
- On UNIX/Linux, to start or stop the server, execute the associated command script as follows:

1. Navigate to the *`server_installation`*`/bin` directory. For example: `/opt/serena/sra/common/tomcat/6.0/bin`
2. Run `startup.sh` to start the server or `shutdown.sh` to stop it.

Starting and Stopping an Agent Relay

If you are using an agent relay and you install the agent relay as a service, the agent relay installer installs the agent relay service but does not start the service. It is set to manual start by default.

Start the agent relay service on your agent relay machine according to the following procedure.

Note: If you are using an agent relay, you must start it before starting any agents that will communicate through it.

To stop or start an agent relay service:

- In Windows, use `Administrative Tools > Services` to start or stop the agent relay of the name you gave it during the installation. The default name is `agentrelay`.

- On UNIX/Linux, to start or stop the agent relay, execute the associated command script as follows:

1. Navigate to the *`agent_relay_installation/bin`* directory. For example: `/opt/Serena/agentrelay/bin`
2. Run `start_agentrelay` to start the server in a new shell. Run `stop_agentrelay` to stop it. To start the agent relay from the command line, run `start_agentrelay`.

Starting and Stopping an Agent

The agent installer installs the agent service, serenaRA Agent (SRA-Agent), but does not start the service. It is set to manual start by default.

Start the agent service on your agent machine according to the following procedure.

Once the agent has started, navigate to the Serena Release Automation web application and display the **Resources** page. If the installation went well, the agent should be listed with a status of `Connected`.

Note: If you are using an agent relay, you must start it before starting any agents that will communicate through it.

To stop or start an agent service:

- In Windows, use `Administrative Tools > Services` to start or stop **serenaRA Agent (SRA-Agent)**.
- On UNIX/Linux, to start or stop the agent, execute the associated command script as follows:

1. Navigate to the *`agent_installation`*`/bin` directory. For example: `../serena/release automation agent/core/bin`
2. Run `start_sraagent` to start the server or `stop_sraagent` to stop it.

Accessing Serena Release Automation

1. Open a web browser and navigate to the host name you configured during installation. For example, `http://localhost:8080/serena_ra/`
2. Log on to the server using the credentials (`Username` and `Password`) you set up during the server installation.

   Once you are logged in, you can change these on the **Settings** tab (see Chapter 11: System Administration [page 109]).
3. Activate the license.

   A license is required in order for the agents to connect to the server. Without a license, Serena Release Automation will be unable to run deployments. For information about acquiring and activating a license, see Licenses [page 109].

Chapter 7: Server Configuration

Depending on your organization's needs, you may configure Serena Release Automation servers as follows:

- *Multiple Servers*: High Availability (HA) active-active configuration that provides the best performance and scalability. (Recommended for most production systems)
- *Two Servers*: Cold Standby, with one active and one passive, or cold, server that activates in case of main server failure.
- *Single Server*: For test and demonstration systems.

Multiple-server High Availability Configuration

For production environments, it is recommended to use the active-active approach to distribute the processing load of agent communication. This lets the agents communicate across multiple servers when many agents are communicating at once, rather than relying on the processing of a single server, providing high server availability and therefore faster response times.

The following describes how to configure two different active-active approaches:

- Agents connect to a single endpoint that uses round-robin DNS or an agent relay (Recommended, truly scalable method). See Connecting Agents to a Single Endpoint [page 76].
- Agents connected to a series of endpoints (Not recommended for large, scalable installations. Recommended more for smaller, test cases). See Connecting Agents to a Series of Server Endpoints [page 76].

Active-active Server Installation

This is the server install process you need to use if you plan on using the active-active feature of Serena Release Automation for server load balancing and fault tolerance.

Note: This install process is a required, prerequisite for the following:

- Connecting Agents to a Single Endpoint [page 76]
- Connecting Agents to a Series of Server Endpoints [page 76]

To install a server for an active-active configuration:

1. Install the shared database (for more information, see Database Installation [page 45]).

2. Install the first Serena Release Automation server into a shared `.serena_ra` folder using a shared database.

   Note: If you use a cluster with a load balancer, you should modify the external agent url, and external user url fields to use the IP address or the DNS name of the cluster (located in the server web interface, Settings tab, System Settings).

3. Install additional servers, making sure to:

   - set the "Install Serena Release Automation to:" field to the shared `.serena_ra` folder.
   - specify the “Use existing settings” option (see (Windows) Server Install: Destination Folder Panel [page 50]).

4. Install agents and then see either Connecting Agents to a Single Endpoint [page 76], or Connecting Agents to a Series of Server Endpoints [page 76].

Connecting Agents to a Single Endpoint

The following steps describe how to connect agents to a single server endpoint.

Important: This configuration is the preferred approach for High Availability (HA) in enterprise environments. In certain circumstances, for example where high availability is required but enterprise level support is not, the alternative approach (see Connecting Agents to a Series of Server Endpoints [page 76] may also be taken.

To connect agents to a single server endpoint:

1. Install the Serena Release Automation server application on *n* servers for the active-active feature (see Active-active Server Installation [page 75]). These *n* servers are used as a cluster with a load balancer.

2. To make it possible for agents to communicate with each of the server nodes within the cluster, during agent installation the IP or DNS-name of the cluster should be set as the communication host.

   In this scenario, the load balancer will choose which of the server nodes communicates with the agent.

Connecting Agents to a Series of Server Endpoints

The following steps describe how to connect agents to a series of server endpoints with which, in the case of a server failure, the agents will attempt to communicate.

Important: In this configuration, high availability is maintained - it is possible to connect agents to a series of endpoints - however enterprise level scalability is not supported. This method is difficult to maintain with a full list of all servers. Use this approach with caution. The preferred approach for High Availability (HA) in enterprise environments is Connecting Agents to a Single Endpoint [page 76]

To configure agents to connect to a single endpoint:

For the purpose of this example, the following procedure assumes a two-node cluster.

Note: The Serena Release Automation server application should be installed on two servers for the active-active feature (see Active-active Server Installation [page 75]). These two servers are *not* used as a cluster with a load balancer.

1. Two example cluster nodes have the following IP addresses: `ip_node_1` and `ip_node_2` (or you can use DNS names).
2. Assuming that during installation, an agent was set to communicate with `ip_node_1`, to enable the agent to also communicate with cluster node `ip_node_2`:
3. In the server web interface, go to the Settings tab and click Network.
4. In the Network tab, click Create New Network Relay.
5. In the Create Network Relay dialog:
   a. Enter a Name for the second server.
   b. In the Host field, enter `ip_node2`
   c. Enter a Port number that the agent will use to communicate with the server.
   d. Check the Active check box.
   e. Click Save.
6. Repeat this process for each cluster node.

Two-server Cold Standby Configuration

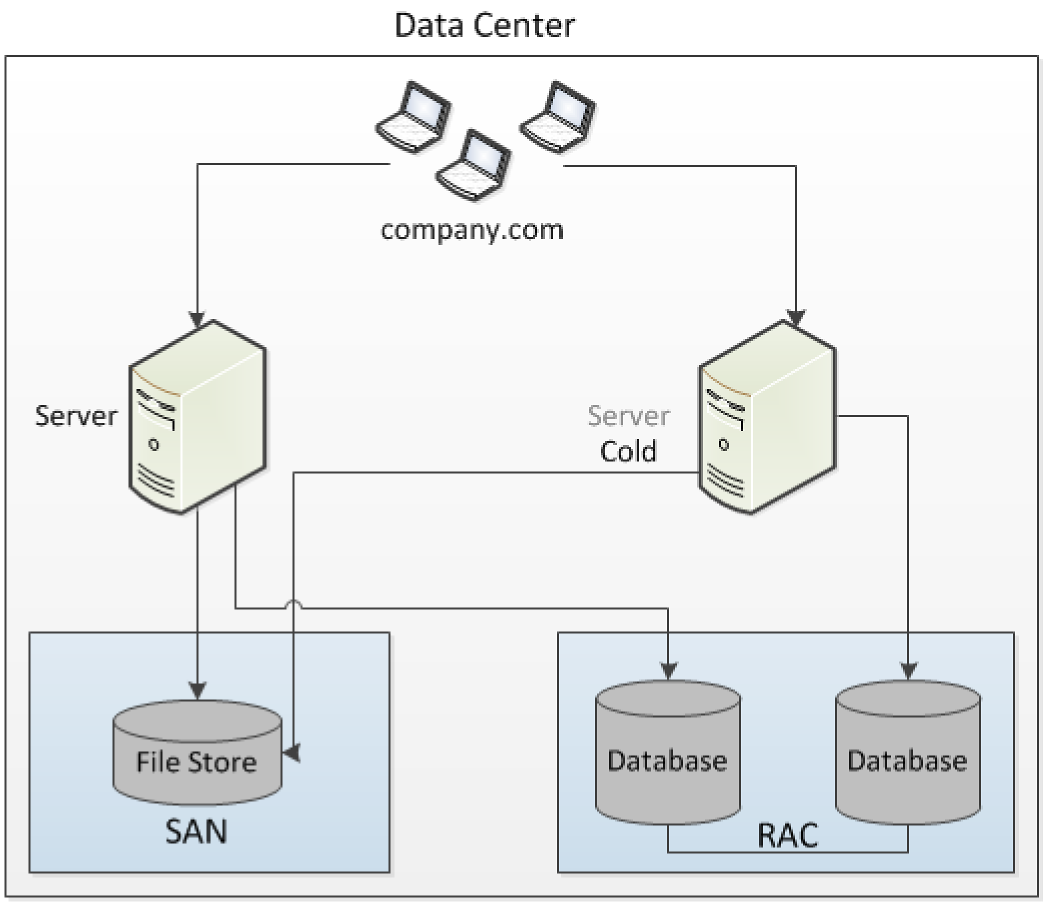
If the active/active load balancing server configuration is not being used, Serena Release Automation employs the cold standby strategy to ensure server availability. When the primary system fails, the cold standby is brought online and promoted to the primary server. Once online, the standby reestablishes connections with all agents, performs recovery, and proceeds with any queued processes. Because the most intense work is handed-off to agents, a high performance configuration should not have an agent installed on the same hardware as the main server.

The Serena Release Automation server aggressively utilizes threading and takes advantage of any additional CPU cores assigned to it. A small to midrange server with 2-4 multi-core CPUs is ideal, but, because it is relatively easy to move an existing Serena Release Automation server installation to a new machine, starting small and scaling as needed is a very legitimate strategy. The memory available to the application tier should also be increased from the default 256 MB to something on the order of 1 GB.

Typical Data Center Configurations

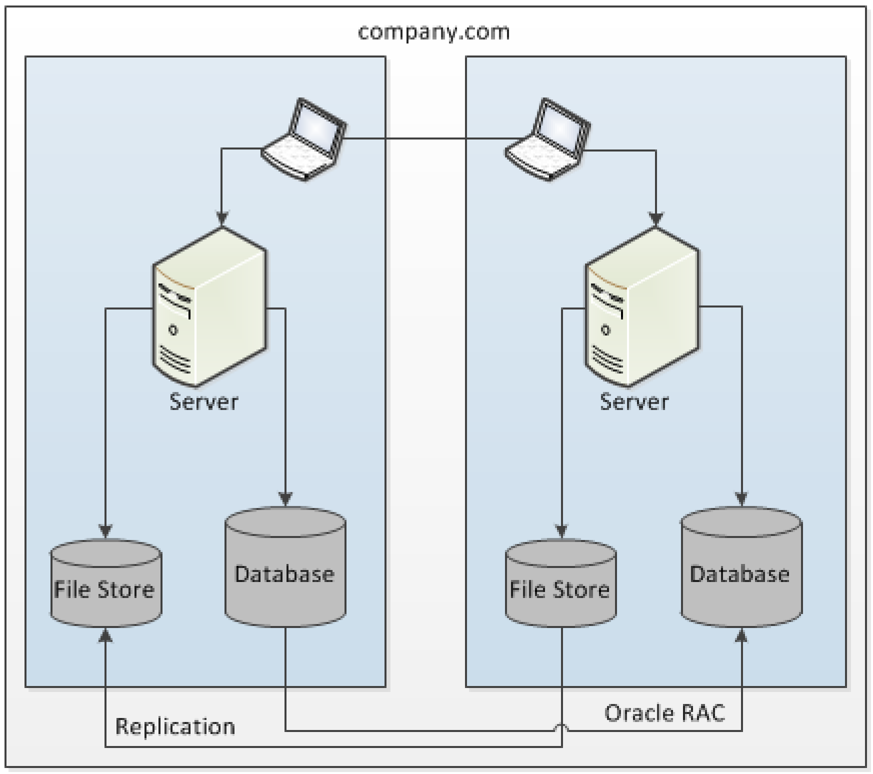
Most organizations configure the data tier with network storage and a clustered database. The service tier performs best when it's on a dedicated, stable, multi-core machine with a fast connection to the data tier. A standby machine should be maintained and kept ready in case the primary server goes down.

Figure 1. Single Data Center Configuration



There are no remote agents or agent relays in this configuration.

Figure 2. Multiple Data Centers



Chapter 8: SSL Configuration

SSL (Secure Socket Layer) technology enables clients and servers to communicate securely by encrypting all communications. Data are encrypted before being sent and decrypted by the recipient–communications cannot be deciphered or modified by third-parties.

Serena Release Automation enables the server to communicate with its agents using SSL in two modes: unauthenticated and mutual authentication. Unauthenticated mode for HTTP and mutual authentication mode for JMS are optional; you can implement one without implementing the other, or implement both.

Configuring Unauthenticated Mode for HTTP Communication

In *unauthenticated mode*, communication is encrypted but users do not have to authenticate or verify their credentials. Serena Release Automation automatically uses this mode for JMS-based server/agent communication (you cannot turn this off).

SSL unauthenticated mode can also be used for HTTP communication. You can implement this mode for HTTP communication during server/agent/agent relay installation, or activate it afterward.

By default Serena Release Automation uses this mode for its JMS-based server/agent communication on port 7918.

Important: Because agent relays do not automatically activate SSL security, you must turn it on during relay installation or before attempting to connect to the relay. Without SSL security active, agent relays cannot communicate with the server or remote agents.

To activate unauthenticated mode for HTTP:

1. Open the `installed.properties` file which is located in the `server_install/conf/server` directory.

   The `installed.properties` file contains the properties that were set during installation.

2. Ensure that the `install.server.web.always.secure` property is set to `Y`.

3. Ensure that the `install.server.web.ip` property is set to the port the server should use for HTTPS requests.

4. Save the file and restart the server.

Note: To complete unauthenticated mode for HTTP, contact Serena Support.

Configuring Mutual Authentication Mode

In *mutual authentication mode*, communications are encrypted as usual, but users are also required to authenticate themselves by providing digital certificates. A digital *certificate* is a cryptographically signed document intended to assure others as to the identity of the certificate's owner. Serena Release Automation certificates are self-signed.

When mutual authentication mode is active, Serena Release Automation uses it for JMS-based server/agent communication. In this mode, the Serena Release Automation server provides a digital certificate to each agent, and each agent provides one to the server. This mode can be implemented during server/agent installation, or activated afterward.

To activate this mode, the Serena Release Automation server provides a digital certificate to each local agent and agent relay, and each local agent and agent relay provides one to the server.

Agent relays, in addition to swapping certificates with the server, must swap certificates with the remote agents that will use the relay. Remote agents do not have to swap certificates with the server, just with the agent relay it will use to communicate with the server.

This mode can be implemented during installation or activated afterward.

Note: When using mutual authentication mode, you must turn it on for the server, agents, and agent relays, otherwise they will not be able to connect to one another. If one party uses mutual authentication mode, they must all use it.

Important: The server and agent properties *must* be set *prior* to configuring mutual authentication and exchanging keys.

To prepare for using mutual authentication, follow these procedures in order:

1. Property Settings for Mutual Authentication [page 82]
2. Adding an Alias to an Agent [page 83]
3. Adding an Alias to an Agent Relay [page 84]
4. Mutual Authentication: Server and Agent(s) [page 84]
5. Mutual Authentication: Server, Agent Relay, and Agent(s) [page 85]

Property Settings for Mutual Authentication

Before you can configure mutual authentication between a server and agent(s), or a server, agent relay and agent(s), you *must* set the following properties.

Table 1. Property Settings Required Before Exchanging Keys table

Property	Location	Value
If not set during install: `server.mutual_auth`	in the server installation directory `.serena\ra\conf\server\installed.properties` file	`true`
`server.jms.mutualAuth`	in the server installation directory `.serena\ra\conf\server\installed.properties` file	`true`

Property	Location	Value
For each agent, if not set during install: agent.mutual_auth	in the agent installation directory [*agent_installdir*]\conf\agent\ →installed.properties file	true
For each agent: locked/agent.mutual_auth	in the agent installation directory [*agent_installdir*]\conf\agent\ →installed.properties file	true
For each agent relay: agentrelay.jms_proxy.secure	in the relay's [*agentrelay_installdir*]\conf\ →agentrelay.properties file	true
For each agent relay: agentrelay.jms_proxy.mutualAuth	in the relay's [*agentrelay_installdir*]\conf\ →agentrelay.properties file	true

Adding an Alias to an Agent

When you install a server, the alias server, to be used for certificate key generation, is assigned to the server. Before you generate a certificate for an agent, you *must* add an alias to the agent.

To add an alias to an agent:

Note: You must have administrative privileges to perform this procedure.

1. Make sure your agent machine JAVA_HOME environment variable points to the directory where java is installed.
2. Open a shell and navigate to the agent install location \conf directory.
3. From within the agent's [*install_location*]\conf directory, run the following command:

```
keytool -genkeypair -dname "cn=[alias_name]" -alias [alias_name]
 -keypass changeit -keystore sra.keystore -storepass changeit
 -keyalg RSA -keysize 1024 -validity 7305
```

   where [*alias_name*] is the name you give the agent alias; we recommend using the agent's name.

4. To check the result, run the command:

```
keytool -list -keystore agent.keystore
```

Adding an Alias to an Agent Relay

Before you generate a certificate for an agent relay, you *must* add an alias to the agent relay.

To add an alias to an agent relay:

Note: You must have administrative privileges to perform this procedure.

1. Make sure your agent relay machine `JAVA_HOME` environment variable points to the directory where java is installed.
2. Open a shell and navigate to the agent relay install location `\conf\jms-relay` directory.
3. From within the agent relay's `[`*`install_location`*`]\conf\jms-relay` directory, run the following command:

   ```
   keytool -genkeypair -dname "cn=[alias_name]" -alias [alias_name]
    -keypass changeit -keystore agentrelay.keystore -storepass changeit
    -keyalg RSA -keysize 1024 -validity 7305
   ```

   where `[`*`alias_name`*`]` is the name you give the agent relay alias; we recommend using the agent relay's name.
4. To check the result, run the command:

   ```
   keytool -list -keystore agentrelay.keystore
   ```

Mutual Authentication: Server and Agent(s)

Make sure your server and agent(s) are not running before you start this configuration.

To configure mutual authentication between a server and agent:

1. Open a shell and navigate to the server install location `\conf` directory:

   ```
   .serena\ra\conf>
   ```

2. Export the server key as a certificate by running:

   ```
   keytool -export -keystore server.keystore -storepass changeit
    -alias server -file server.crt
   ```

   You should see the message:

   ```
   Certificate stored in file server.crt
   ```

3. Copy the exported `server.crt` (certificate file) to the agent `[`*`install_location`*`]\[`*`agent_name`*`]\conf` directory.

   where `[`*`agent_name`*`]` is the unique name you gave the agent during install.

4. From within the agent's `[install_location]\[agent_name]\conf` directory, import the `server.crt` file by running:

   ```
   keytool -import -keystore sra.keystore -storepass changeit -alias server
    -file server.crt -keypass changeit -noprompt
   ```

   You should see the message:

   ```
   Certificate was added to keystore
   ```

5. From within the agent's `[install_location]\[agent_name]\conf` directory, export the agent key as a certificate by running:

   ```
   keytool -export -keystore sra.keystore -storepass changeit
    -alias [agent_alias] -file [agent_name].crt
   ```

   The certificate is stored in the `[agent_name].crt` file.

   Important: Before you export an agent key, you *must* first add an alias to the agent (see Adding an Alias to an Agent [page 83]).

6. Copy the exported `[agent_name].crt` (certificate file) to the server installation location `\conf` directory.

7. From within the server's install location `.serena\ra\conf` directory, import the `[agent_name].crt` file by running:

   ```
   keytool -import -keystore server.keystore -storepass changeit
    -alias [agent_alias] -file [agent_alias].crt -keypass changeit -noprompt
   ```

   You should see the message:

   ```
   Certificate was added to keystore
   ```

8. For additional agents, repeat from step 5.

9. Start the server and agent(s).

Mutual Authentication: Server, Agent Relay, and Agent(s)

These instructions are for configuring mutual authentication for a server, agent relays, and agents that communicate with the server through agent relay(s). Make sure your server and agent(s) are not running before you start this configuration.

To configure mutual authentication between a server, agent relay and agent(s):

1. Open a shell and navigate to the server install location `\conf` directory:

   ```
   .serena\ra\conf>
   ```

2. Export the server key as a certificate by running:

```
keytool -export -keystore server.keystore -storepass changeit
 -alias server -file server.crt
```

You should see the message:

```
Certificate stored in file server.crt
```

3. Copy the exported `server.crt` (certificate file) to the agent relay `[install_location]\[agentrelay_name]\conf\jms-relay` directory.

   where `[agent-relay_name]` is the unique name you gave the agent relay during install.

4. From within the agent relay directory:

   ```
   [install_location]\[agent-relay_name]\conf\jms-relay
   ```

   import the `server.crt` file by running:

   ```
   keytool -import -keystore agentrelay.keystore -storepass changeit
    -alias server -file server.crt -keypass changeit -noprompt
   ```

   You should see the message:

   ```
   Certificate was added to keystore
   ```

   Important: Before you can export an agent relay key, you *must* first add an alias to the agent relay (see Adding an Alias to an Agent Relay [page 84]).

5. From within the agent relay directory:

   ```
   [install_location]\[agent-relay_name]\conf\jms-relay
   ```

   export the agent relay key as a certificate by running:

   ```
   keytool -export -keystore agentrelay.keystore -storepass changeit
    -alias [agent-relay_alias] -file [agent-relay_name].crt
   ```

   You should see the message:

   ```
    Certificate is stored in file [agent-relay_name].crt
   ```

6. Copy the exported `[agent-relay_name].crt` (certificate file) to the server installation location `.serena\ra\conf` directory.

7. From within the server's install location `.serena\ra\conf` directory, import the `[agent-relay_name].crt` file by running:

   ```
   keytool -import -keystore server.keystore -storepass changeit
    -alias [agent-relay_alias] -file [agent-relay_name].crt
    -keypass changeit -noprompt
   ```

You should see the message:

```
Certificate was added to keystore
```

8. For an agent that is configured to connect to the agent relay, copy the exported `[agent-relay_name].crt` (certificate file) to the agent `[install_location]\[agent_name]\conf` directory.

   where `[agent_name]` is the unique name you gave the agent during install.

9. From within the agent's `[install_location]\[agent_name]\conf` directory, import the `[agent-relay_name].crt` file by running:

```
keytool -import -keystore sra.keystore -storepass changeit
 -alias [agent-relay_alias] -file [agent-relay].crt
 -keypass changeit -noprompt
```

   You should see the message:

```
Certificate was added to keystore
```

10. From within the agent's `[install_location]\[agent_name]\conf` directory, export the agent key as a certificate by running:

```
keytool -export -keystore sra.keystore -storepass changeit
 -alias [agent_alias] -file [agent_name].crt
```

   You should see the message:

```
 Certificate is stored in file [agent_name].crt
```

   Important: Before you can export an agent's certificate key, you *must* first add an alias to the agent (see Adding an Alias to an Agent [page 83]).

11. Copy the exported `[agent_name].crt` (certificate file) to the agent relay `[install_location]\[agent-relay_name]\conf\jms-relay\` directory.

12. From within the agent relay `[install_location]\[agent-relay_name]\conf\jms-relay` directory, import the `[agent_name].crt` file by running:

```
keytool -import -keystore agentrelay.keystore -storepass changeit
 -alias [agent_alias] -file [agent_name].crt -keypass changeit -noprompt
```

   You should see the message:

```
Certificate was added to keystore
```

13. To configure another agent that communicates with the server through this agent relay, repeat from step 10.

   Attention: For each agent, make sure you change the name of the `–alias` argument `[agent_alias]`, and the `–file` argument `[agent_name]`.

14. Restart the server, agent relay, and agents.

Chapter 9: Automation Administration

This documentation contains the following automation administration sections:

- Automation Plug-ins [page 89]
- Locks [page 89]
- Post-Processing Scripts [page 91]
- Statuses [page 91]

Automation Plug-ins

Plug-ins can be installed at any time. You can download a zip file that contains all the plug-ins from http://www.serenasupport.com.

To download the plug-ins:

1. Go to http://support.serena.com/Case/CaseHome.aspx and log in using your customer account.
2. Browse to the My Downloads tab.
3. From the Please Select Product drop-down, select `Serena Release Automation`.
4. Download the plug-ins zip file to the platform on which you want to deploy it.

Note: You *do not* need to decompress the .zip file.

To install the downloaded plug-ins:

1. From the Automation Plug-ins pane, display the Load Plug-in dialog `Automation > Automation Plugins > Load Plugin [`*`button`*`]`.
2. Enter the path to the compressed plug-in (.zip) file and click Submit.

After the plug-in load process completes successfully, the plug-ins are listed on the Automation Plug-ins pane. Once installed, plug-in functionality is available immediately.

Locks

A *lock* is a routinely used to ensure that processes do not interfere with one another. Normally, once a lock is no longer needed it is released. Sometimes a lock will not get released and its associated process will be unable to complete. The lock management feature enables you to quickly identify and resolve abnormal lock conditions.

Managing Locks

A running process with a lock, like all active processes, appears on the Dashboard page with a status of `Running`. If a locked process takes longer to complete than expected, you can cancel the process from the Dashboard, or investigate it fully with the `Settings` tab.

1. Display the Lock pane by clicking the `Locks` link on the Settings tab `Administration > Automation > Locks`.

   The Lock pane displays the following information:

   Lock Fields table

Field	Description
Name	The name identifies the lock. The displayed name is a concatenation of the component or application name (depending on type) + process name + resource name.
Type	Indicates whether the process creating the lock is a component- or application-type. Locks can only be applied to component or application processes.
Component/ Application	Displays the name of the component or application containing the lock. Clicking an item displays (depending on the type) the Component pane, or Application pane, where you can investigate the lock.
Resource/ Environment	Displays the name of the resource or environment containing the lock. Clicking an item displays (depending on the type) the Resource pane, or Environment pane.
Process	Displays the name of the process containing the lock. Clicking an item displays the process in the process editor.
Actions	Lists the available actions.

2. Resolve the lock by selecting an action:

   Lock Actions table

Action	Description
View Request	Displays the process log for the process containing the lock. You can use the `Actions` field on the displayed pane to see the name of the process step causing the lock.
Release	Releases the lock which enables the associated process to continue processing.

If the Serena Release Automation server and or agents go down while a locked process is running, Serena Release Automation will automatically restore any interrupted processes along with any locks they might contain once service is restored.

Post-Processing Scripts

Serena Release Automation component processes perform post-processing whenever a plug-in step finishes execution. Typically, post-processing scripts ensure that expected results occurred. You can use your own JavaScript script instead by instructing Serena Release Automation to use your script when you define the step, see Process Editor [page 135].

When a step finishes, the agent performing the step will run your script (the script must be written in JavaScript). When the agent runs the script, it first loads the server log file and finds the exit code property of the target step using regular expressions defined in the script. It then applies any actions defined in the script before processing the next step.

To create a script:

1. Display the **Edit Script** dialog `Automation > Post Processing Scripts`.
2. Enter a name for the script into the `Name` field. The name must match the name you specified when you defined the process step, (see Process Editor [page 135]).
3. Enter or paste the script into the `Script Body` field. See the roll-over help next to the field for information about the properties and variables available for user-defined scripts.

The Serena Release Automation server log file is normally found in the following location: `Serena Release Automation_root\var\log\deployserver.out`.

Statuses

The *Inventory Statuses* page shows what applications and components have been deployed, including the current versions that are running on the resource within an environment. The inventory provides complete visibility into the different versions of your applications which can be tracked back to the original artifacts imported into Serena Release Automation.

There are different views of the current inventory, depending on where in Serena Release Automation you are. Inventory information is available on the individual components, for every application environment, as well as for each resource (agent).

Resources Inventory

The Resources Inventory, `Resources > [select resource] > Inventory`, tells you what components are on an environment. From here, selecting either the component or its version will take you to the component's page if you need more information.

Component Inventory

The Component Inventory, `Components > [select component] > Inventory`, tells you what version of the component is running on a resource.

For example, if the component is currently deployed to multiple machines, they would all be displayed. From here, you can navigate to the resource.

Environment Inventory

The Environment Inventory, `Application > [select application] > Environments`, tells you what version of any given component is running on a particular resource for each application environment. If multiple versions are running on different resources, they are all listed.

Chapter 10: Security Administration

Serena Release Automation provides a flexible, Roles and Permissions [page 94] that maps to your organizational structure. Different product areas, such as components, can be secured by roles. Each area has a set of permissions available to it. To configure security for an area, you create roles using the available permissions—execute, read, write, and so forth.

So, how are permissions applied to users? First, global default permissions can be granted. Default permissions are granted by product area and apply to all users. If default permissions are granted for, say, the agent area, a user will have those permissions even if she is also part of a group or role that does not.

Groups

Another way users can be granted permissions is by being a member of a group. Groups can have default permissions that apply to all group members. If a user is assigned to a group with default permissions for the agent area, as above, she will have those permissions even if she is also assigned a role that does not have them.

Roles and Permissions

Finally, users can be assigned to roles. Role members inherit a role's permissions. Except for UI and system security, users are assigned to roles on an item by item basis. For example, a user can be assigned a role that enables them to see only one application or only one component. Both groups and individual users can be assigned to roles.

Roles and permissions, including default permissions, are configured on an area by area basis; granting the execute permission to one role does not grant it to another. The default admin role has all permissions, but you can create another user with all permissions by creating a role for each area with all permissions granted, then assigning the user to each role. Typically, new roles are added to product areas during setup and occasionally thereafter.

While any number of roles can be created for an area, areas themselves cannot be created, modified (the available pool of permissions cannot be changed), or deleted.

Related Topics

- Setting up Security [page 93]

Setting up Security

Generally, you perform the following steps in order when setting-up Serena Release Automation security:

1. **Create Roles**

   Create roles and define permissions for the various product areas (see Creating and Editing Roles [page 95]). For most evaluations, the default roles should be adequate.

Use the UI security area to quickly assign access permissions to the different areas of Serena Release Automation.

Use the system security area to assign usage permissions, including the ability to define security for other users.

2. **Authorization Realms**

   Authorization realms are used by authentication realms to associate users with groups and to determine user access (see Authorization Realms [page 101]). Serena Release Automation includes both an internal database for storing security information as well as integration with the Lightweight Directory Access Protocol (LDAP). LDAP is a widely-used protocol for accessing distributed directory information over IP networks. If you are implementing a production version of Serena Release Automation, the LDAP integration is recommended. If you are evaluating Serena Release Automation, it is not necessary to set up the LDAP integration—full security is configured and enforced by the server.

3. **Create Groups and Define Default Permissions**.

   Determine default permissions by product area. Global default permissions can be granted.

4. **Create Authentication Realm**

   The authentication realm is used to determine a user's identity within an authorization realm. If more than on realm has been configured, user authentication is determined following the hierarchy of realms defined on the Authentication pane. When a user attempts to log in, all realms are polled for matching credentials.

5. **Add Users**

   Add users to an authentication realm, then assign them to groups and roles. If your are using LDAP, you can import users and map them to the security system.

Roles and Permissions

Roles provide the building blocks for the security system. Roles have permissions that define the actions the roles can perform with product features. Typical actions include changing or executing an item, such as an application process, or modifying its security settings. Users or groups assigned to a role are automatically granted the permissions configured for it. The default roles can be edited and new roles can be created.

Serena Release Automation maps key product features or areas to security roles. Each area has several permissions defined for it (listed below). When you create a role, you first specify the product area. Selecting a product area defines the set of permissions available to the new role—only permissions defined for the area are available.

Generally, permissions fall into one of these groups:

Common Permissions table

Permission	Description
Security	Enables users to change an item's security settings. For example, a user with this permission for agents can determine which users can view, configure, and set security for them.
Write	Enables users to add, change, and delete items. A user with this permission for components can create a component.
Read	Enables users to read (view) an item, but not change it or create another of its type. A user with this permission for agents, say, will be able to see agents within the user interface, but will not be able to modify them or create another unless granted additional permissions.
Execute	Enables users to run processes associated with applications, components, environments, and resources. Users must also have read permission for an item before actually executing it.

Default Roles

Serena Release Automation ships with several role types mapped to product areas. Every area or type has a set of available permissions. The *application* type, for instance, has the Manage Snapshots permission in addition to the common permissions (see the Roles and Permissions [page 94] table. User-defined roles within a type can choose from among the permissions available for that type.

Every product area has one role typically called `Admin` or `Administrator` that has all permissions available for that area. Deleting a default Admin role for one role type does not affect the Admin role for another type.

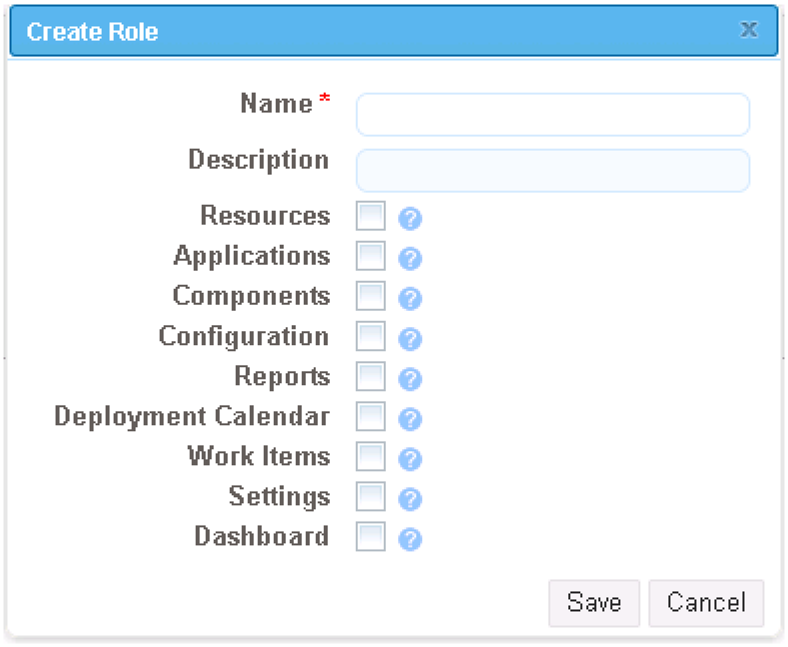
You can quickly grant a role type's permissions to all users using the Default Permissions tab.

Note: Default permissions cannot be granted for system and UI security.

Creating and Editing Roles

1. Display the Role Configuration pane `Security > Role Configuration`.
2. From the list of product areas, select the area where you want to add a role.
3. Click `Create Role [`*`button`*`]`.

Figure 1. Web UI Role Permissions



All permissions available for this area display.

4. Select the permissions you want granted to this role.

All roles have the following permissions available. Other permissions, if any, are described in the following sections.

Table 1. Permissions Available for Every Role

Permission	Description
Security	Manage security for the effected feature area.
Write	Create, edit, or delete items for this product area.
Read	Access or view items for this product area.

Agent Roles

Agent roles define the functions users can perform with agents and agent pools. Available permissions are read, write, and security, (see Creating and Editing Roles [page 95]).

To add users to agent roles:

1. Display the Security tab for the target agent `Resources > Agents/Agent Pools > [`*`selected agent/agent pool`*`] > Security`).

   All roles defined for agents and agent pools are displayed.

2. Use the Add Role Member action for a specific role, then select the user.

All users are available. As shipped, Serena Release Automation provides an `Admin` role with all configured permissions granted. By default, `Admin` has a single user, `admin`.

Application Roles

Application roles define the functions users can perform with applications. In addition to the standard permissions (see Creating and Editing Roles [page 95], others are:

Application Roles table

Permission	Description
Manage Snapshots	Create and edit snapshots for this application.
Run Component Processes	Run associated component processes outside of the application.

To add users to application roles:

1. Display the Security tab for the target application (see Chapter 15: Managing Applications [page 155]) `Applications > [`*`selected application`*`] > Security`).

   All defined roles are displayed.

2. Use the Add Member action for a specific role, then select the user.

   All users are available. As shipped, Serena Release Automation provides an `Admin` role with all configured permissions granted. By default, `Admin` has a single user, `admin`.

Component Template Roles

These roles define the functions users can perform with component templates. Available permissions are read, write, and security (see Creating and Editing Roles [page 95].

To add users to component template roles:

1. Display the Security tab for the target template `Components > Templates > [`*`selected template`*`] > Security` (see Component Templates [page 148]).

   All defined roles are displayed.

2. Use the Add Role Member action for a specific role, then select the user.

   All users are available. As shipped, Serena Release Automation provides an `Admin` role with all configured permissions granted. By default, `Admin` has a single user, `admin`.

Component Roles

These roles define the functions users can perform with components. In addition to the standard permissions, others are available:

Component Roles table

Permission	Description
Manage Versions	Create and delete versions for this component.

To add users to component roles:

1. Display the Security tab for the target component (see Creating Components [page 122]) `Components > [`*`selected component`*`] > Security`.

   All defined roles are displayed.

2. Use the Add Role Member action for a specific role, then select the user.

   All users are available. As shipped, Serena Release Automation provides an `Admin` role with all configured permissions granted. By default, `Admin` has a single user, `admin`.

Environment Roles

These roles define the functions users can perform with environments (see the Creating and Editing Roles [page 95]

To add users to environment roles:

1. Display the Security tab for the target environment `Components > [`*`selected component`*`] > Security`.

   All defined roles are displayed.

2. Use the Add Role Member action for a specific role, then select the user.

   All users are available. As shipped, Serena Release Automation provides an `Admin` role with all configured permissions granted. By default, `Admin` has a single user, `admin`.

License Roles

These roles define the functions users can perform with licenses. Available permissions are read, write, and security (see the Creating and Editing Roles [page 95]).

To add users to license roles:

1. Display the Security tab for licenses (see Licenses [page 109]) `Security > Role Configuration > Licenses`.

   All defined roles are displayed.

2. Use the Add Role Member action for a specific role, then select the user.

   All users are available. As shipped, Serena Release Automation provides an `Admin` role with all configured permissions granted.

   By default, `Admin` has a single user, `admin`.

Resource Roles

These roles define the functions users can perform with resources. Available permissions are read, write, execute, and security, (see Creating and Editing Roles [page 95].

To add users to resource roles:

1. Display the Security tab for the target resource `Security > Resources > [`*`selected resource`*`] > Security)`. For resource groups: `Security > Default Permissions > Resource Groups > [`*`Edit Group action`*`] > Security`.

   All defined roles are displayed.

2. Use the Add Role Member action for a specific role, then select the user.

   All users are available. As shipped, Serena Release Automation provides `Admin` role with all configured permissions granted. By default, `Admin` has a single user, `admin`.

System Security

These roles define the functions users can perform with the Serena Release Automation server (also referred to as system security). Available permissions are:

Server Permissions table

Permission	System Settings Description
Security	Manage security configuration; users without this permission cannot access or change the security functions.
Manage Plug-ins	Grants users the ability to install new plug-ins (see Automation Plug-ins [page 89]).
Create Subresources	Ability to create subresources (see Chapter 16: Managing Resources [page 173]).
Create\Manage Resource Roles	Create and delete resource roles (see Resource Roles [page 174]).
Create Components	Create components (see Creating Components [page 122]).
Create Applications	Create applications (see Creating Applications [page 156]).
Create Component Templates	Create component templates (see Creating a Component Template [page 149].
Manage Licenses	Add and remove licenses (see Licenses [page 109]).

To add users to system security roles:

1. Display the System Security tab `Administration > Security > System Security`.

   All defined roles display.

2. Use the **Add Role Member** action for a specific role, then select the user.

   All users are available.

   As shipped, Serena Release Automation provides `Configuration Manager` and `System Administrator` roles; the latter has all configured permissions granted. By default, `System Administrator` role has a single group— `Admin Group` (with user admin), and the `Configuration Manager` role also has a single group— `Configuration Group` (with user config).

User Interface Security

These roles determine which parts of the Serena Release Automation web application users can access. Each page, such as Reports, on the web application's home page can be restricted. Available permissions are:

Web UI Permissions table

Permission	Description
Resources	Access the Resources page.
Applications	Access the Applications page.
Components	Access the Components page.
Configuration	Access the Configuration page.
Reports	Access the Reports page.
Deployment Calendar	Access the Calendar page.
Work Items	Access the Work Items page.
Settings	Access the System Settings page.
Dashboard	Access the Dashboard page.

To add users to Web UI roles:

1. Display the System Security page `Administration > Security > Security)`.

   For resource groups: `Resources > Resource Groups >` [*`Edit Group action`*] `> Security`.

   All defined roles are displayed.

2. Use the Add Role Member action for a specific role, then select the user.

All users are available. As shipped, Serena Release Automation provides the following roles:

Default Web UI Roles table

Role	Description
Deployment Engineer	Access the Reports, Calendar, Work Items, and Dashboard pages.
Approver	Access the Reports, Work Items, and Dashboard pages.
Administrator	Access all pages.
Configuration Engineer	Access all pages except *Calendar* and *Work Items*.

Authorization Realms

The Authorization Realms pane is used to create user groups and authorization realms. Authorization realms associate users with roles and work with authentication realms to determine which users can access Serena Release Automation. The authorization realms available are:

Internal Storage

Uses internal role management. The default authorization realm— `Internal Security`—is this type.

LDAP

Uses external LDAP role management.

Creating an LDAP Authorization Realm

An LDAP authorization realm uses an external LDAP server for authorization.

To create an LDAP authorization realm:

1. Display the Create Authorization Realm dialog `Administration > Security > Authorization (Groups) > Authorization Realms > Create Authorization Realm [`*`button`*`]`.
2. Ensure that `LDAP` is selected in the Type list box, then specify the following:

   LDAP Authorization Realm Properties table

Field	Description
User Group Attribute	Name of the attribute that contains role names in the user directory entry. If user groups are defined in LDAP as an attribute of the user, the Group Attribute configuration must be used

Field	Description
Group Search Base	Base directory used to execute group searches, such as ou=employees,dc=mydomain,dc=com.
Group Search Filter	LDAP filter expression used when searching for user entries. The name will be substituted in place of 0 in the pattern, such as uid={0}. If this is not part of the DN pattern, wrap the value in parenthesis, such as ud=(0).
Group Name	Directory name used to bind to LDAP for searches, such as cn=Manager,dc=mycompany,dc=com. If not specified, an anonymous connection will be made. Required if the LDAP server cannot be anonymously accessed.
Search Group Subtree	Searches the subtree for the roles if checked.

Creating Authorization Groups

Groups are logical containers that serve as a mechanism to grant permissions to multiple users; members automatically share a group's permissions (see Default Permissions [page 103]). Permissions are granted to groups (or all users), not individual users. Additionally, when a group is assigned a role, its members are automatically assigned the role as well (see Roles and Permissions [page 94]).

To create a group:

1. Display the Create Group dialog `Administration > Security > Authorization (Groups) > Groups > Create Group [`*`button`*`]`.
2. Provide a name for the group.

   The name appears in the Default Permissions pane (see Default Permissions [page 103]).
3. Select an authorization realm.

   Groups are only valid for the selected realm.

Serena Release Automation provides several default groups and users, which are listed in the following table. The default groups and users are part of the internal security authorization realm.

Default Groups table

Group	Users
Admin Group	admin
Approve Group	approve

Group	Users
Configuration Group	config
Deploy Group	deploy

Default Permissions

Default permissions can be set globally for all users for a product area, or for individual user groups within an area. By default, a product areas' permissions are *not* enabled for any user or group (except for the `admin` user which has all permissions for all role types granted). Use the Default Permissions page to set default permissions, for both the groups you create and those shipped with the product.

Users added to a group inherit the group's default permissions.

Setting Default Permissions

To set default permissions:

1. Display the Default Permissions pane `Security > Default Permissions`.
2. From the list of product areas, select the area you want to use.

   Selecting an area displays the permissions available for it. User-defined groups are configured independently (see Authorization Realms [page 101]).
3. Check the permissions you want to grant for the selected group.

   The following table lists the available permission.

   Product Area Privileges table

Role	Read	Write	Security	Execute	Snapshots	Comp. Procss.	Versions
Agent	X	X	X				
Agent Pool	X	X	X				
Application	X	X	X	X	X	X	
Component	X	X	X	X			X
Component Template	X	X	X				
Environment	X	X	X	X			
License	X	X	X				
Process	X	X	X	X			

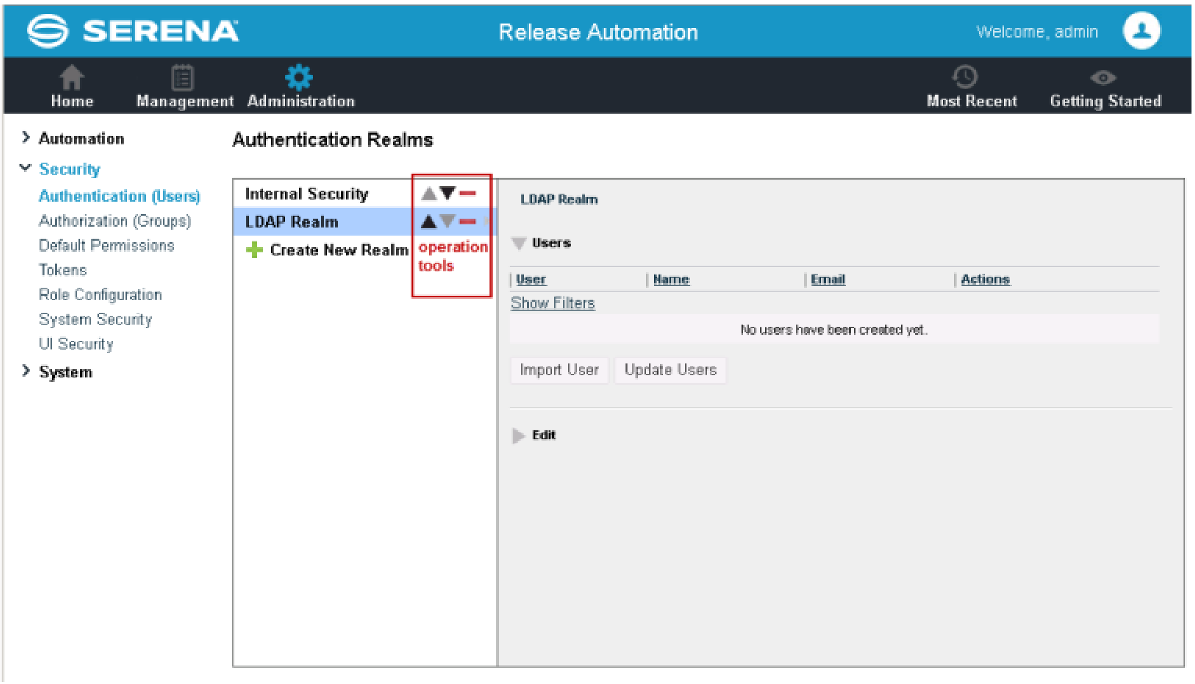
Role	Read	Write	Security	Execute	Snapshots	Comp. Procss.	Versions
Resource	X	X	X	X			
Resource Group	X	X	X	X			

Authentication Realms

The Authentication Realms pane, `Security > Authentication (Users)`, is used to create authentication realms and users. Authentication realms determine a user's identity within an authorization realm. Authentication is determined following the hierarchy of realms displayed on the Authentication Realms pane.

In the following example, authentication will first be determined in the Internal Security realm followed by the LDAP realm. A user listed in the LDAP realm may have different authorizations from those in the other realms.

Authentication Realms Precedence



No users have been created yet.

If you have a number of authentication realms, you can reorder them using the operation tools. Each realm can be moved up to a higher priority, moved down, or deleted by using the operation tools.

Creating an Authentication Realm

1. Display the Create New Authentication Realm `Administration > Security > Authentication (Users) > Authentication Realms > Create New Realm [`*`button`*`]`.

2. Enter a name and description and other basic parameters:

Allowed Login Attempts	Number of attempts allowed. A value of 0 means unlimited attempts.
Authorization Realm	Requires that the authorization realm was previously created.
Type	Selecting `Internal Storage` completes the process.

Creating an LDAP Authentication Realm

If you create an `LDAP` type of authentication realm, you need to provide information about your LDAP installation:

LDAP Authentication Realm Properties table

Field	Description
Context Factory	Context factory class used. This may vary depending upon your Java implementation. The default for Sun Java implementations: `com.sun.jndi.ldap.LdapCtxFactory`.
LDAP URL	URL to the LDAP server beginning with `ldap://` or `ldaps://`. Separate additional servers with spaces.
Use DN Pattern	User directory entry pattern; the name will be substituted in place of 0 in the pattern, such as cn={0},ou=employees,dc=yourcompany,dc=com.
User Search Base	Base directory used to execute group searches, such as ou=employees,dc=mydomain,dc=com.
User Search Filter	LDAP filter expression used when searching for user entries. The name will be substituted in place of 0 in the pattern, such as uid={0}. If this is not part of the DN pattern, wrap the value in parenthesis, such as ud=(0).
Search User Subtree	If the LDAP user names are case sensitive, check the box to treat different-case names as different users.
Search Connection DN	Directory name used to bind to LDAP for searches, such as cn=Manager,dc=mycompany,dc=com. If not specified, an anonymous connection will be made. Required if the LDAP server cannot be anonymously accessed.
Search Connection Password	Password used when connecting to LDAP to perform searches.

Field	Description
Name Attribute	Contains the user's name, as set in LDAP.
Email Attribute	Contains the user's email address, as set in LDAP.

Once configuration is complete, when a new user logs on using their LDAP credentials, they will be listed on the Authentication Realm Users pane. It is best practice not to manage user passwords nor remove users from the list. If an active user is removed from Serena Release Automation, they will still be able to log onto the server as long as their LDAP credentials are valid.

Authentication Realm Users

When adding a new user, the user name and password is what the individual will use when logging into Serena Release Automation. The user name will also be displayed when setting up additional security.

Once the new user has been successfully added to a group, you might need to configure additional permissions. This can happen when the new user is mapped to a group that has limited permissions.

Importing LDAP Users

Unless using LDAP authorization realm, valid LDAP users can log on but will have no permissions. To provide permissions, import them first and define their permissions before they log on. You can import users from existing LDAP systems into Serena Release Automation-managed authentication realms.

To Import LDAP Users

1. Display the Create User dialog `Administration > Security > Authentication (Users) > Authentication Realms > [`*`select LDAP realm`*`] > Import User`.
2. Enter the name of the user.

   If you enter a search filter in the `Username` field, the filter must be enclosed in parentheses.

Tokens

Tokens provide authorization for agents and users. Agents use tokens when performing process steps and communicating with the Serena Release Automation server and external services. Users can use tokens with the CLI client (see Chapter 22: Command Line Client (CLI) [page 223]), and instead of supplying a user name and password in certain situations.

You can create tokens in addition to those shipped with the product.

To create a token:

1. Display the Create New Token dialog `Administration > Security > Tokens > Create New Token [`*`button`*`]`.
2. From the User drop-down list box, select the user who will use the token.
3. Specify the expiration date and time.

Tokens can be used immediately after being created.

Chapter 11: System Administration

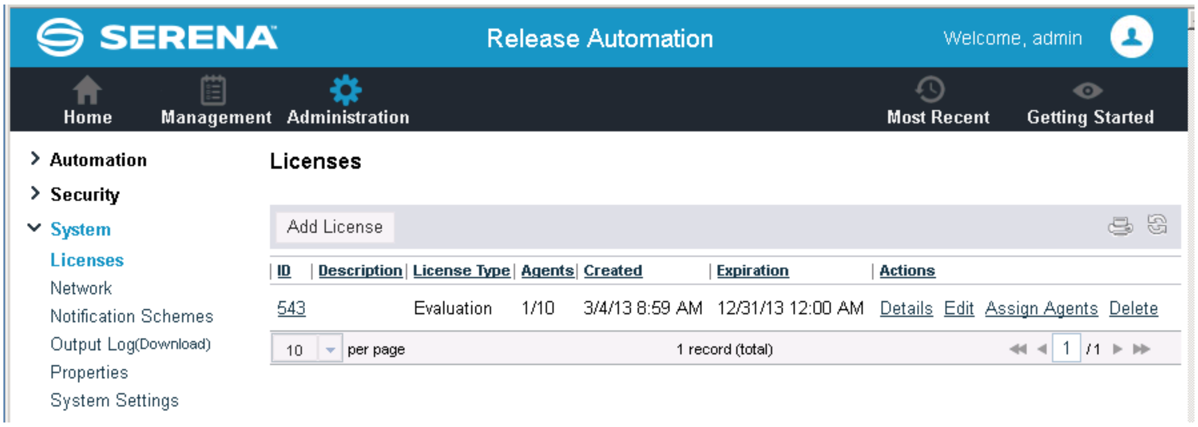
You can use the Administration System options of the Web client to access, create, and/or modify features such as:

- licensing
- downloading the output log
- configuring network relays
- specifying system settings

Licenses

The Licenses pane is where you manage user licenses (adding or deleting licenses, and assigning agents to them). Display the Licenses pane by clicking the Licenses link on the Settings window `Administration > System > Licenses`. You can also access the pane through the `Resources` pane `Management > Resources > License`.

Licenses Pane



ID	Description	License Type	Agents	Created	Expiration	Actions
543		Evaluation	1/10	3/4/13 8:59 AM	12/31/13 12:00 AM	Details Edit Assign Agents Delete

10 per page   1 record (total)   1 /1

Adding a License

To add a license:

1. Display the Add New License dialog by clicking the Add New License button.
2. Paste the license text supplied by Serena into the License field.
3. (Optional) Add a description.
4. Click Save when you are done.

To see information about a license, display the License Details pop-up by clicking the Details link.

Adding Agents to a License

Agents can be assigned to licenses automatically or manually. This section explains how to add agents manually. To automatically add agents, ensure that the Automatic License Management check box on the System Settings pane is checked (see Chapter 11: System Administration [page 109]).

To add an agent to a license manually:

1. Display the Assign Agents to License pop-up by clicking the Assign Agents link for the intended license.
2. Click the Agents field.

   A selection-type pop-up displays.
3. Make a selection from the list of agents that are not already assigned to the license.
4. Select the agent or agents you want to add to the license.
5. Optional. You can filter the listed agents by entering search text into the text field.
6. After select agents, click OK to close the selection pop-up.
7. If you want to restart the selection process, click Reset.
8. When you are finished, click Save.

Modifying or Deleting a License

To modify or update an existing license:

1. Display the Edit License dialog by clicking the Edit link for the license you want to change.
2. Edit the information shown in the License field.
3. Click Save.

To delete an existing license, click the Delete link for the selected license.

Network Relay

A *network relay* is used in conjunction with an agent relay. The network relay reverses the direction of communication through a firewall between the Serena Release Automation server and agent relay. A network relay is only used when you want the server to connect to the relay instead of the reverse (which is default). To create a network relay an agent relay must be created. (See Installing Agent Relays [page 69] to create an agent relay.)

To create a network relay:

1. Display the network pane `Administration > System > Network > Create New Network Relay [button]`.
2. Enter the name of the network relay.
3. Identify the Host and Port.

4. Indicate the Network Relay will be Active by checking the box.

Notifications

Serena Release Automation can send email notifications whenever user-defined trigger events occur. Notifications can be sent when a deployment finishes or an approval is required, for example. Notification recipients are defined with the security system's (see Chapter 10: Security Administration [page 93]) LDAP integration. If you have not already done so, set up LDAP prior to configuring notifications. Serena Release Automation relies on LDAP and an associated e-mail server to send notifications.

Note: Serena Release Automation requires an external SMTP mail server to send notifications. For information about configuring a mail server, see Chapter 11: System Administration [page 109].

When setting up notifications, you select both the triggering events and the role, which is inherited from the security system, to determine which users will receive notification. For example, it is common for an administrator or environment owner to be notified when a work item (as part of the approval process) has been generated. The default notification scheme, which sends notifications to the application and admin default roles (see Chapter 10: Security Administration [page 93]), can be edited or you can create your own scheme.

To set up your own notifications, display the Notifications pane `Administration > System > Notification Schemes`.

Configure the new scheme. Here, you will be setting up the who/when for notifications. Once configured, you can come back add additional entries to the scheme or edit an existing one.

Notification Type

To select notification types to send for a particular role, navigate to the **Add Notification Entry** pop-up, `Administration > System > Notification Schemes > [select scheme] > Add Notification Entry [button]`, and select from the **Type** drop-down menu.

The process type is determined mainly by the type of recipient. For example, a deployment engineer would be interested in being notified about a failed deployment (Process Failure).

Notification Target

To select notification targets for a particular role, navigate to the **Add Notification Entry** pop-up, `Administration > System > Notification Schemes > [select scheme] > Add Notification Entry [button]`, and select from the **Target** drop-down menu.

When setting the target, the application option will only send out notifications when the event selected above corresponds to an Application. For example, the "Process Success" event, when paired with the "Application" Target would trigger a notification when a Process (an application deployment) is successful. Similarly, the same event type, when used with the "Environment" target would instigate a notification when a successful deployment has been run in an Environment, such as SIT or PROD).

Notification Role

To select a role to be notified, navigate to the **Add Notification Entry** pop-up, `Administration > System > Notification Schemes > [select scheme] > Add Notification Entry [button]`, and select from the **Role** drop-down menu.

The Role corresponds to those set in the Security System. Any individual assigned the Role you select will receive an e-mail.

Template Name

To select a notification template name, navigate to the **Add Notification Entry** pop-up, `Administration > System > Notification Schemes > [select scheme] > Add Notification Entry [button]`, and select from the **Template Name** drop-down menu.

The available templates are provided by default and should meet all your needs, including formatting the email being sent. Which template you use is based on why you want to set up a notification and the recipients of the notification. However, if the default templates do not suit your needs, you can create your own.

Template	Description
Application deployment failure/success	Sends notifications about a specific application to the specified users, based on the role setting.
Task readied/created/completed	Used to report back on the state of manual tasks.
Deployment readied	A specialized email template for letting people know a deployment has been prepared.
Approval created/failed	These templates are used to notify the status of an approval.

Once you have the entry done, add others using the same process. If you want to use the new notification scheme with existing applications, modify the application settings.

Related Topics

- Creating Notification Templates [page 112]

Creating Notification Templates

Notification Templates are XML files located in the server’s `conf/server/notification-template` file folder. If the default notification templates do not suit your needs, you can create new ones.

To create a new notification template:

1. Start a new XML file.
2. Enter Script. (Notification templates only supports Velocity Reports)
3. Save file in the server's `conf/server/notification-template` file folder.

4. Restart the server.

Related Topics

- Notifications [page 111]

Output Log

You can download the Serena Release Automation server log from within the web application.

To download the log file:

1. Display the Output Log dialog by clicking the `Output Log` link on the Settings pane.
2. Click Download Log to save the file.
3. Optional. You can download the file directly from the Settings pane by clicking the `Download` link on the Settings pane.

The Serena Release Automation server log file default location is: `Serena Release Automation_root\var\log\deployserver.out`.

System Settings

The following system settings can be specified in `Administration > System > System Settings`.

Setting	Description
External Agent URL	URL for agents to access the Serena RA web UI.
External User URL	URL for users to access the Serena RA web UI.
Only Groups in Security Roles	When this is enabled, individual users cannot be granted permissions to anything. All permissions must be granted to groups.
Automatic Version Import Check Period (seconds)	The number of seconds between polls for new versions for all components with this option turned on. Requires a server restart to take effect. It is not recommend to set this to less than 15.
Mail Server Host	Hostname of the mail server for notifications. Leave this blank to disable notifications.
Mail Server Port	SMTP port for email notifications.

Setting	Description
Secure Mail Server Connection	Whether the SMTP connection should be secured.
Mail Server Sender Address	Address to use for the "from" address for email notifications.
Mail Server Username	Username to use for sending email notifications.
Mail Server Password	Password to use for sending email notifications.
Hour to Clean Versions	The hour of the day when versions should be cleaned up. This must be an integer between 0 (midnight) and 23 (11 PM).
Days to Keep Versions	Number of days to keep component versions. -1 will keep indefinitely.
Number of Versions to Keep	Number of versions to keep for each component. -1 will keep all.
Archive Path	Path to write a zip containing the artifacts of each version cleaned up. If this is blank, archives will not be written.
Minimum Password Length	The minimum length of passwords allowed to be created.
Require Complex Passwords	This will validate new passwords created through the UI or command line interface to ensure the password is complex. To pass validation, it must use 2 of the 4 character classes: upper, lower,digits, and special. It must be as long or longer than the password minimum length.
Preview Version Cleanup	Preview the component versions that will be archived the next time an archive file is created.
Automation License Management	Whether the server should automatically assign new agents to open license slots.

Chapter 12: File Versioning Repository

The following topics explain the architecture of the file versioning repository in Serena Release Automation.

- Codestation [page 115]
- Relocating Codestation [page 116]

Codestation

File versioning in Serena Release Automation is handled by Codestation. You can direct Serena Release Automation to introduce artifacts into Codestation from the file system or from external source control tools that you identify when you select the **Source Config Type** for a component.

Codestation artifacts represent deployable items such as files, images, databases, configuration materials, or anything else associated with a software project. By default, these are stored in the `var` subdirectory in the Serena Release Automation server installation directory.

Serena Release Automation's secure and tamper-proof artifact repository ensures that deployed components are identical to those tested in pre-production environments. Without the repository, artifacts would have to be pulled from network shares or some other system, increasing both security risks and the potential for error.

The artifact repository uses content addressable storage to maximize efficiency while minimizing disk use. The repository tracks file versions and maintains a complete history for all components. Maximizing efficiency is important, since the artifact repository stores files that are much larger than source files.

Association of files with Components is built into the system. Without any configuration, each Component gets its own area of the repository for its files. There is no chance of confusion or mix-up of files to Components. And, each Component Package is mapped to a specific set of files and versions corresponding to the Component.

The artifact repository comes with a client application that provides remote access to the repository. Using the client, the user can add/modify files, create Packages, retrieve files, as well as view the history of changes.

The client application provides a file transfer capability that can be used to deliver files to target servers during deployments. This built-in transfer mechanism verifies the integrity of all transferred files against their expected cryptographic signatures, thus guaranteeing that files have not been corrupted during transmission or tampered with during storage.

In addition to the client application, the artifact repository exposes REST-based web services. These services are used to build integrations between build systems such as AnthillPro and Serena Release Automation. Such integrations automatically place the artifacts produced by the build process in the artifact repository, thus making the artifacts available for deployment.

In an enterprise environment, the default installation might not be ideal. See Relocating Codestation [page 116] for a discussion about enterprise options.

Relocating Codestation

By default, the data tier's log files and Codestation artifacts are stored in the `var` subdirectory within the Serena Release Automation server directory. Ideally, this data should be stored on robust network storage that is regularly synchronized with an off-site disaster recovery facility. In addition, the Serena Release Automation server should have a fast network connection to storage (agents do not need access to the storage location). In Unix environments, you can use *symbolic links* from the `var` subdirectory to network storage. On Windows platforms there are several options for redirecting logs and artifacts, including `mklink` (supported in Windows 7 and later).

If you want to relocate Codestation, relocate both the `var` directory as well as the `\logs\store` directory. A good formula for determining Codestation storage requirements is: `average artifact size * number of versions imported per day * average number of days before cleanup`

Distributed teams should also take advantage of Serena Release Automation location-specific Codestation proxies to improve performance and lower WAN usage.

Chapter 13: Integrating Release Automation with Serena Solutions

You can integrate Serena Release Automation functionality into your Serena Business Manager (SBM) solutions using SBM Composer REST service support.

This supports:

- Design Mode in SBM Composer
- Import of Serena Release Automation WADL into SBM REST Grids
- Return of Serena Release Automation data directly from SBM (RESTfully)

Note: For more information on using the SBM Composer REST service support, please see the Serena Business Manager (SBM) documentation and the Serena Support Knowledgebase.

Part 3: Using Serena Release Automation

This section contains the following information:

- Chapter 14: Managing Components [page 121]
- Chapter 15: Managing Applications [page 155]
- Chapter 16: Managing Resources [page 173]
- Chapter 17: Configuration and Processes Options [page 179]
- Chapter 18: Deployments [page 181]
- Chapter 19: Reports [page 185]
- Chapter 20: Example—helloWorld Deployment [page 205]
- Chapter 21: Serena Release Automation Properties [page 219]
- Chapter 22: Command Line Client (CLI) [page 223]

Chapter 14: Managing Components

You create and configure components in the Web client user interface as detailed in the following sections.

- Component Creation Overview [page 121]
- Creating Components [page 122]
- Component Versions [page 131]
- Component Processes [page 134]
- Process Editor [page 135]
- Configuring Access for Process Steps [page 145]
- Component Manual Tasks [page 147]
- Post-Processes [page 148]
- Component Templates [page 148]
- Deleting and Deactivating Components [page 153]

Component Creation Overview

Components are the centerpiece of Serena Release Automation's deployment engine. Components associate items that will be deployed—artifacts—with processes that will deploy them. The following table summarizes the basic steps performed to create components. Related topics are listed below the table.

Component Creation Steps table

Step	Description
1. Define source configuration	Define the source type and identify the artifacts associated with the component. The source type can be any or nearly any associated with a software project. Once defined, all artifacts must be of the defined type, see Creating Components [page 122].
2. Create component version	Create the initial component version by importing artifacts into the artifact repository, CodeStation. Versions can be imported manually or automatically. Version imports can be full (all artifacts are imported) or incremental (only changed artifacts are imported). Serena Release Automation tracks all artifact changes which enables you to rollback components or deploy multiple versions of the same one.

Step	Description
3. Create component process	Use the process design editor to create a process for the component. Component processes consist of user-configured steps that operate on the component, usually by deploying it. The available steps are provided by installed plug-ins. As shipped Serena Release Automation provides plug-ins for many common functions. Numerous other plug-ins are available from Serena—http://support.serena.com.

Related topics:

- How to create manual tasks, see Component Manual Tasks [page 147]
- How to install plug-ins, see Automation Plug-ins [page 89]
- How to create and use templates, see Creating a Component Template [page 149]
- How to import component templates, see Importing Templates [page 150]

Creating Components

In general, component creation is the same for all components.

To create a component:

1. Navigate to the **Create Components** dialog `Components > Create Component` *[button]*. Several fields are the same for every source, while others depend on the source type selected with the **Source Config Type** field.
2. Define standard parameters. The basic fields are available for every source type.

   See Basic Fields [page 122].
3. If you select a value in the **Source Config Type** and **Import Versions Automatically** fields, additional selections are available. Enter values into the source-specific fields. See Source Configuration Fields [page 124] for information about the source configuration types.
4. When finished, save your work. Saved components are listed in the Component pane.

Basic Fields

These fields appear for all source types; they are displayed when the Create Component dialog opens. Other fields, discussed below, are displayed when a source type is selected.

Fields Available for All Source Types table

Field	Description
Name	Identifies the component; appears in many UI features. Required.

Field	Description
Description	The optional description can be used to convey additional information about the component. If the component is used by more than one application, for example, entering "Used in applications A and B" can help identify how the component is used.
Template	A component template enables you to reuse component definitions; components based on templates inherit the template's source configuration, properties, and process. Any previously created templates are listed. A component can have a single template associated with it. The default value is `None`. If you select a template, the Template Version field is displayed which is used to select a template version. By controlling the version, you can roll-out template changes as required. The default value is Latest Version which means the component will automatically use the newest version (by creation date). See Component Templates [page 148]. Note If you select a template that has a source configured for it, the dialog box will change to reflect values defined for the template. Several fields, including the Source Config Type field, will become populated and locked.
Source Config Type	Defines the source type for the component's artifacts; all artifacts must have the same source type. Selecting a value displays additional fields associated with the selection. Source-dependent fields are used to identify and configure the component's artifacts. If you selected a template, this field is locked and its value is inherited from the template.

Field	Description
Import Versions Automatically	If checked, the source location is periodically polled for new versions; any found are automatically imported. • Polling Period The time period in seconds between each poll for new versions. The polling interval is controlled by the Automatic Version Import Check Period (seconds) field on the Settings pane (`Administration > System > System Settings`. The default value is 15 seconds. • Quiet Period The time period in minutes in which no polling is done. The quiet period starts after the last polling period that detected changes. There is no default quiet period; you must set it for each source type. If left unchecked, you can manually create versions by using the Versions pane. By default, the box is unchecked. See Importing Versions Automatically [page 133] for more information.
Copy to CodeStation	This option—selected by default—creates a tamper-proof copy of the artifacts and stores them in the embedded artifact management system, CodeStation. If unchecked, only meta data about the artifacts are imported. Serena recommends that the box be checked.
Default Version Type	Required. Defines how versions are imported into CodeStation. `Full` means the version is comprehensive and contains all artifacts; `Incremental` means the version contains a subset of the component's artifacts. Default value is: Full.
Inherit Cleanup Settings	Determines how many component versions are kept in CodeStation, and how long they are kept. If checked, the component will use the values specified on the System Settings pane. If unchecked, the Days to Keep Versions (initially set to -1, keep indefinitely) and Number of Versions to Keep (initially set to -1, keep all) fields are displayed, which enable you to define custom values. The default value is checked.

Source Configuration Fields

Depending on the Source Config Type you select, various tool-specific fields appear. Click the ? icon next to each field for more information.

The source configuration tools that you can select include the following:

[/concept/conbody/p/simpletable {"nocellnorowborder"})

• None • AnthillPro • ClearCaseUCM • Serena Dimensions CM [page 126] • File System (Basic) [page 125] • File System (Versioned) [page 126] • Git • Jenkins • Luntbuild • Maven	• Serena PVCS [page 127] • Perforce • StarTeam • Subversion • TFS • TFS_SCM • TeamCity • TeamForge • uBuild

(simpletable]

File System (Basic)

Imports everything in the target directory whenever you import versions. You can set up a template to auto-increment version numbers. Automatic import is not available for this source type.

File System (Basic) Source Fields table

Field	Description
Base Path	Defines how the property is presented to users in the Serena Release Automation editor. This element has several attributes.
Always Use Name Pattern	Used to specify values for a select-box. Each value has a mandatory `label` attribute which is displayed to users, and a value used by the property when selected. Values are displayed in the order they are defined.
Version Name Pattern	Defines how the property is presented to users in the Serena Release Automation editor. This element has several attributes.
Next Version Number	Defines how the property is presented to users in the Serena Release Automation editor. This element has several attributes.
Save File Execute Bits	Defines how the property is presented to users in the Serena Release Automation editor. This element has several attributes.

File System (Versioned)

The File System (Versioned) source type interacts with file-system-based artifacts. It assumes that subdirectories within the base directory are distinct component versions. File System (Versioned) can automatically import versions into CodeStation.

File System (Versioned) Source Fields table

Field	Description
Base Path	Path to directory containing artifacts. The content of each subdirectory within the base directory is considered a distinct component version. The subdirectory with the most recent time-stamp is considered the "latest version".
Save File Execute Bits	Defines how the property is presented to users in the Serena Release Automation editor. This element has several attributes:

Serena Dimensions CM

Serena Dimensions CM is a software configuration management tool. To use Serena Dimensions CM as an artifact source, select `Dimensions` from the Source Config Type drop-down list box then configure the type-specific fields described here. For information about creating components, see Creating Components [page 122].

See Basic Fields [page 122] for information about the standard fields which apply to each source type.

Serena Dimensions Fields table

Field	Description
Username	Dimensions CM user name. For information about user impersonation, see User Impersonation [page 145].
Password	Password associated with the Dimensions user name.
DB Name	Name of the Dimensions database.
DB Connection	Name of the Dimensions connection to be used. A connection/session is required in order to send or receive commands to/from the database.
Server	Server managing the Dimensions database.
Product Spec	Location of the Dimension-managed artifacts.

Serena PVCS

Serena PVCS is a version control management tool. To use Serena PVCS as an artifact source, select `PVCS` from the Source Config Type drop-down list box then configure the type-specific fields described here. For information about creating components, see Creating Components [page 122].

See Basic Fields [page 122] for information about the standard fields which apply to each source type.

Serena PVCS Fields table

Field	Description
PCLI Path	Path to the PVCS CLI tool.
Database Path	Path to the PVCS database.
Base Path	Base path of the repository.
Project Path	Path to the project.
Archive Path	The location of the archive relative to the database path.
Includes	The patterns to match files to upload. The wild card ** indicates every directory and the wildcard * indicates every file. So, the pattern `dist/**/*` would retrieve the entire file tree underneath the `dist` directory.
Excludes	The patterns to exclude files to upload.
User	Serena PVCS username. For information about user impersonation, see User Impersonation [page 145]
Password	Password associated with the Serena PVCS username.
Always Use Name Pattern	If this option is selected, the Version Name Pattern specified will always be used. Not selected by default.
Version Name Pattern	An optional template for automatic version names. Use `${version}` to reference the Next Version Number field.
Next Version Number	An integer used for the next version created. The number is automatically incremented for each version created.

Field	Description
Preserve Execute Permissions	To save file execute permissions with the files, select this check box. Not selected by default.

Component Properties

The component properties available are: custom, environment, and version. Another type, component, is defined by template and becomes part of any component created from the template, see Component Template Properties [page 150]. Property versions, or changes, are maintained and remain available.

The three types can be defined on the component's Properties pane `Management > Components > [`*`selected component`*`] > Properties`. The three types are described in the Component Properties table.

Component Properties table

Property Type	Description
Properties	Custom property; can be used in scripts and plug-ins. Properties inherited from templates cannot be modified on the component level.

Property Type	Description
Environment	Available to environments that use the component. The property will appear on the environment's Component Mappings pane Applications > [*selected application*] > Environments > *selected environment*] > Component Mappings, see Application Environments [page 160]. Each property must have a type: • *Text* Enables users to enter text characters. • *Text Area* Enables users to enter an arbitrary amount of text, limited to limited to 4064 characters. • *Check Box* Displays a check box. If checked, a value of true will be used; otherwise the property is not set. • *Select* Requires a list of one or more values which will be displayed in a drop-down list box. Enables a single selection. **Note:** Not currently implemented. • Multi Select Requires a list of one or more values which will be displayed in a drop-down list box. Enables multiple selections. • Secure Used for passwords. Similar to *Text* except values are redacted.
Version	Available to every component version Management > Components > [*selected component*] > Versions > [*selected version*] > Properties. Values can be set at the individual version level. Each property must have a type (described above).

Importing/Exporting Components

Components can be imported and exported. Importing/exporting can be especially useful if you have multiple Serena Release Automation servers, for example, and need to quickly move or update components.

Exporting Components

Exporting a component creates a JSON file (file extension `json`) that contains the component's source configuration information, properties, and processes. For information about JSON, see www.json.org.

To export a component:

On the Components pane `Management > Components`, click the `Export` link in the Actions field beside the component that you want to export. You can load the file into a text editor, or save it. If you save it, a file is created with the same name as the selected component, for example, `helloWorld.json`.

Importing Components

When you import a component, you can create an entirely new component or upgrade an existing one. Additionally, if the imported component was created from a template, you can use it or create a new one.

Note: If the imported component has the Import Versions Automatically parameter set to true, the new component will automatically import component versions as long as the artifacts are accessible to the importing server.

To Import a Component

1. Display the Import Component dialog `Management > Components > Import Component [`*`button`*`]`.
2. Enter the path to the JSON file containing the component definition or use the Browse button to select one.
3. If you want to upgrade an existing component, check the Upgrade Component check box. To create a new component, leave the box unchecked.

   If the component's name in the JSON file (not the name of the file itself) matches an existing component, the component's parameters are updated with the new values, and new items—such as processes—are added. If the name of the component is not found, the command has no effect.

   Note: The component's name is the first parameter in the JSON file; for example,

   ```
   "name": "helloWorld",
   ```

4. If the imported component was originally created from a template, use the Component Template Upgrade Type drop-down box to specify how you want to use the template. For these options, the template must be on the importing server. If the imported component was not created from a template, these options are ignored.
   - To use the imported component's template, select Use Existing Template. The new component will be an exact copy of the imported one and contain a pointer to the imported component's template. This option is especially useful if you are importing a lot of components based on the same template.

     If you are upgrading, the component will also point to the imported template.

- To create a new template, select Create Template. The new component will be an exact copy of the imported one and contain a pointer to the newly created template (which is based on the imported component's template).

  If you are upgrading a component, a new template is also created used.

- When you want to create a fresh installation and ensure a template is *not* on the importing server, select Fail if Template Exists. If you are creating a component, it will create both a new component and template unless the template already exists, in which case the component is not imported.

  If you are upgrading a component, the upgrade will fail if the imported component's template already exists.

- To ensure the template is on the importing server, select Fail if Template Does Not Exist. If you are creating a component, it will create both a new component and template unless the template does not exist, in which case the component is not imported.

  If you are upgrading a component, the upgrade will fail if the imported component's template does not exist on the importing server.

- To upgrade the template, select Upgrade if Exists. This option creates a new component and upgrades the template on the importing server. If the template does not exist, a new one is created.

5. Click Submit.

Component Versions

Each time a component's artifacts are imported into the repository, including the first time, the component is versioned. Versions can be assigned automatically by Serena Release Automation, applied manually, or come from a build server. Every time a component's artifacts are modified and reimported, a new version of the component is created. So a component might have several versions in CodeStation and each version will be unique.

A version can be full or incremental. A full version contains all component artifacts; an incremental version only contains artifacts modified since the previous version was created.

Importing Versions Manually

To import versions manually, use the following procedure.

1. Display the Version pane for the component you want to use `Components > [`*`select component`*`] > Versions`.

   All versions statuses come from active/inactive `Source Config Type` field.

2. Enter `helloWorld` in the Name field.

   Display the Import.

3. Enter a description in the Description field.

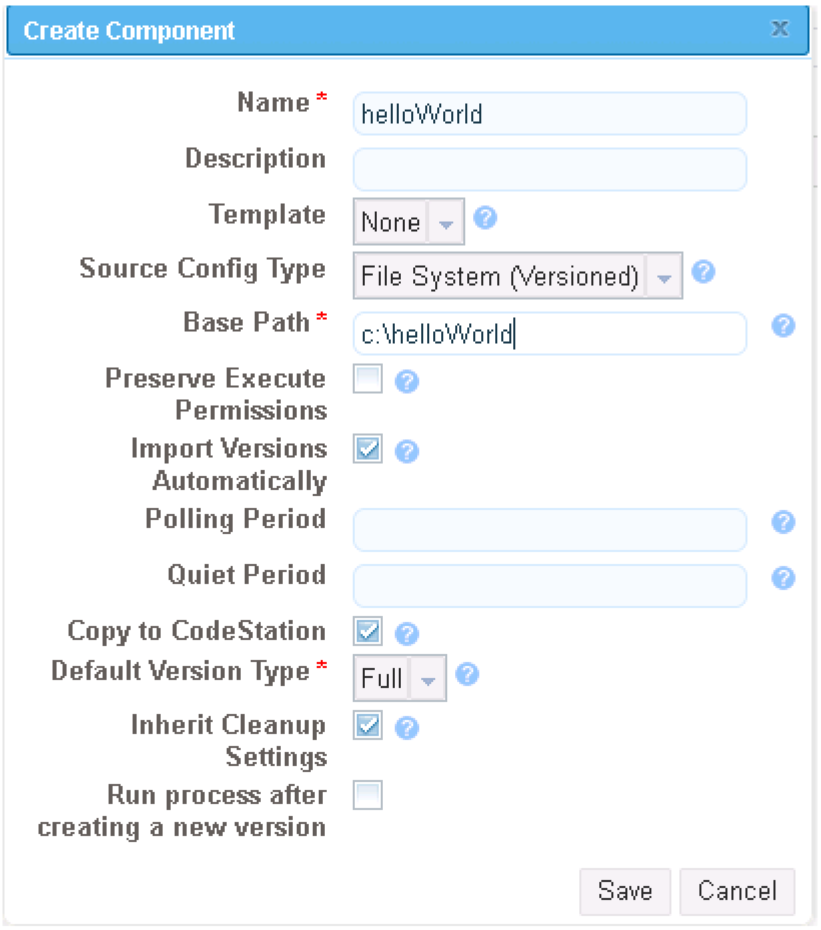
   The optional description can be used to convey additional information about the component. If the component is used by more than one application, for example, entering "Used in applications A and B" can help identify how the component is used. If you are unsure about what to enter, leave the field blank. You can always return to the component and edit the description at any time. In an attempt to appear hip, I entered *Euro store* for my component.

For this exercise, ignore the Template field. Templates provide a way to reuse commonly used component configurations. For information about templates, see Component Templates [page 148].

4. Select `File System (Versioned)` from the Source Config Type field.

   Selecting a value displays several fields required by the selected type.

   Source Config Type



Create Component

Name *	helloWorld
Description	
Template	None
Source Config Type	File System (Versioned)
Base Path *	c:\helloWorld
Preserve Execute Permissions	☐
Import Versions Automatically	✓
Polling Period	
Quiet Period	
Copy to CodeStation	✓
Default Version Type *	Full
Inherit Cleanup Settings	✓
Run process after creating a new version	☐

   `File System (Versioned)` is one of the simplest configuration options and can be used to quickly set-up a component for evaluation purposes, as we do here.

5. Complete this option by entering the path to the artifacts.

   In our example, the artifacts are stored inside the subdirectory created earlier. `File System (Versioned)` assumes that subdirectories within the base directory are distinct component versions, which is why we placed the files (artifacts) inside a subdirectory. `File System (Versioned)` can automatically import versions into

CodeStation. If a new subdirectory is discovered when the base directory is checked, its content is imported and a new version is created.

6. Check the Import Versions Automatically check box.

   When this option is selected, the source location will be periodically polled for new versions. If this option is not selected, you will have to manually import a new version every time one becomes available.

Importing Versions Automatically

When this option is selected, the source location is periodically polled for new versions; any found are automatically imported. The default polling period is 15 seconds, which can be changed with the System Settings pane, see Chapter 11: System Administration [page 109]).

You can also set a quiet period. Specified in minutes, the quiet period gives time for any changes detected in a particular polling period to be versioned before the next polling period starts. This prevents any changes done during processing to be missed in the next polling period.

Component Version Statuses

Component version statuses are user-managed values that can be added to component versions. Once a status is added to a version, the value can be used in component processes or application gates (see Application Gates [page 170]).

Version statuses can be applied to a component version though the user interface `Components > [`*`selected component`*`] > Versions > [`*`selected version`*`] > Add a Status [`*`button`*`]`, or by the Add Status to Version plug-in step.

Serena Release Automation-provided statuses are defined in an XML file which you can freely edit to add your own values (see Structure of the default.xml File [page 171]).

Deleting Component Versions

You can delete any component version. To delete a version, use the Delete action for the version `Components > [`*`selected component`*`] > Versions > [`*`selected version`*`] > Delete`. Deleting a version removes associated meta data from the repository; original artifacts are unaffected.

Inactivating Component Versions

Inactive component versions remain in the repository (unlike deleted versions) but cannot be deployed. To render a component version inactive, use the Inactivate action for the version `Components > [`*`selected component`*`] > Versions > [`*`selected version`*`] > Inactivate`.

To make an inactive version active, use the Show Inactive Versions check box and the Activate action.

Component Change Logs

Change logs provide information about modifications to components. To see change details, display the log for a selected component-related activity `Management >`

`Components > Changes [`*`selected component`*`] > Changes > Changes [`*`action for selected item`*`]`. Information for any change that triggered a Commit ID is displayed.

Component Processes

A component process is a series of user-defined steps that operate on a component's artifacts. Each component has at least one process defined for it and can have several.

Component processes are created with Serena Release Automation's process editor. The process editor is a visual drag-and-drop editor that enables you to drag process steps onto the design space and configure them as you go. Process steps are selected from a menu of standard steps.

Serena Release Automation provides steps for several standard utility processes such as inventory management and workflow control. Additional process steps are provided by plug-ins.

Serena Release Automation provides plug-ins for many common processes, such as downloading and uploading artifacts, and retrieving environment information. See the *Serena Release Automation Plug-in Guide* for more information.

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

Configuring Component Processes

A component process is created in two steps: first, you configure basic information, such as name; second, you use the process editor to configure the process.

To configure a component process:

1. Display the Create New Process dialog `Management > Components > [select component] > Processes > Create New Process [button]`.
2. The dialog's fields are described in the following table.
   Table 1. Create New Process Fields table

Field	Description
Name	Identifies the process; appears in many UI elements. Required.
Description	The optional description can be used to convey additional information about the process.

Field	Description
Process Type	Required. Defines the process type. Available values are: <ul><li>**Deployment:** deploys a component version to the target resource and updates the inventory after a successful execution.</li><li>**Configuration Deployment:** configuration-only deployment with no component version or artifacts—simply applies the configuration (using properties, or configuration templates) to the target agent and updates the resource configuration inventory afterwards.</li><li>**Uninstall:** standard uninstall that removes a component version from a target resource and the resource's inventory.</li><li>**Operational (With Version):** operational process which does not add or remove any artifacts or configuration; runs arbitrary steps given a component version. Useful when you want to start or stop some service for a previously deployed component version.</li><li>**Operational (No Version Needed):** same as the previous type, but does not require a component version.</li></ul>
Inventory Status	Required. Status applied to component versions after being successfully executed by this process. `Active` indicates the component version is deployed to its target resource; `Staged` means the component version is in a pre-deployment location. The status appears on the Inventory panes for the component itself and environments that ran the process.
Default Working Directory	Required. Defines the location used by the agent running the process (for temporary files, etc.). The default value resolves to *agent_directory*\work\component_name_directory. The default properties work for most components; you might need to change it if a component process cannot be run at the agent's location.
Required Component Role	Restricts who can run the process. The available options are derived from the Serena Release Automation security system. The default value is `None`, meaning anyone can run the process. For information about security roles, see Chapter 10: Security Administration [page 93].

3. Save your work when you are finished. The process is listed on the Processes pane for the associated component.

Process Editor

After creating a process, use the process editor to design the process.

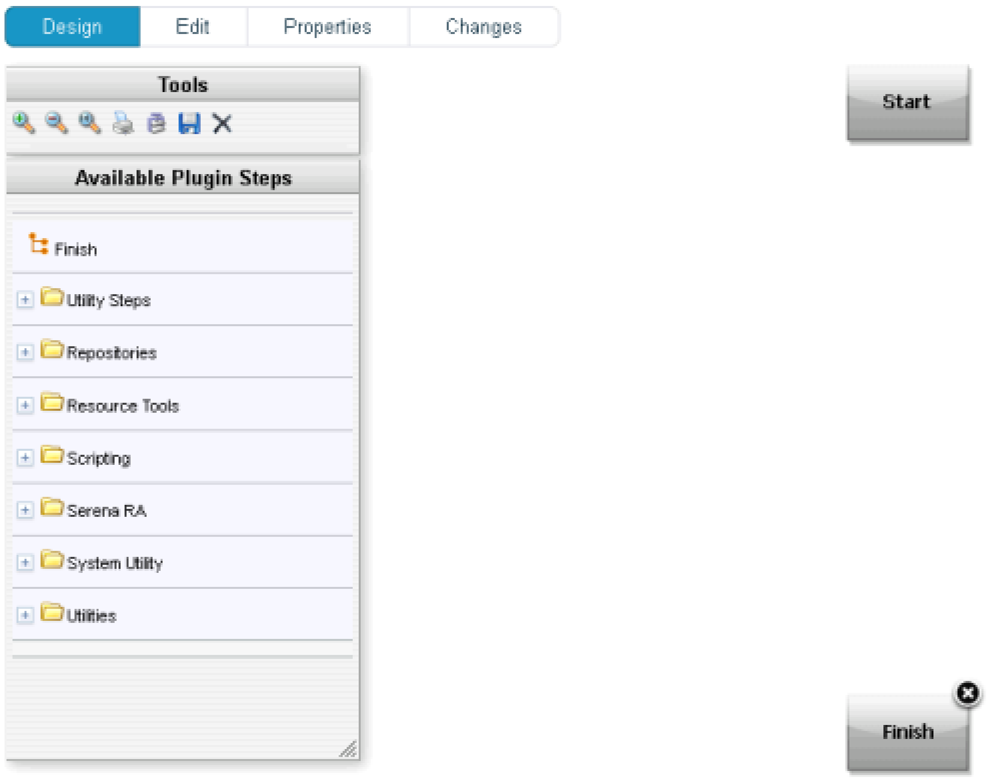
Displaying the Process Editor

To access the process editor:

1. Select the process you want to design.
2. Select the **Design** tab.

   The **Design** pane is displayed.

   Process Design Pane



Available steps are listed in the **Available Plug-in Steps** list. See the *Serena Release Automation Plug-in Guide* for more information on plug-ins.

Using the Process Editor

When the **Process Design** pane opens, the **Design** view displays. Processes are configured on the **Design** view. Several other views can be displayed by clicking the associated tab:

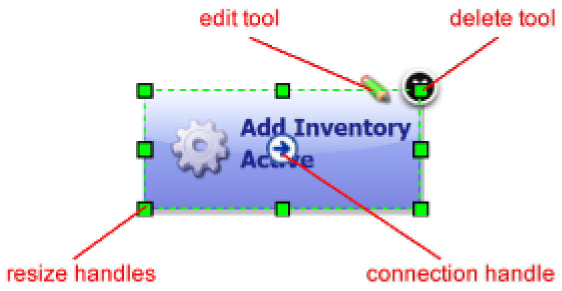
Available Views table

Tab/ View	Description
Edit	Displays the **Edit** view where you can change process parameters.

Tab/ View	Description
Properties	Displays the **Properties** view where you can create and change process properties.
Changes	Displays the **Process Changes** view. This view provides a record for every process change–property add or delete, and process save or delete.

Processes are configured by dragging individual steps onto the design space, configuring, and then connecting them. When a step is dragged onto the design space, a pop-up displays that is used to configure the step. Once configured and the pop-up closed, you define relationships between steps by dragging *connection handles* between associated steps.

Typical Process Step



Graphically, each step (except for the Start step which cannot be deleted or edited) is the same and provides:

Anatomy of a Step table

Item	Description
edit tool	displays the step configuration pop-up where you can modify configuration parameters
delete tool	removes the step from the design space
resize handle	enables you to resize the step graphic
connection tool	used to create connections between steps

Note: If you delete a step, its connections (if any) are also deleted.

Adding Process Steps

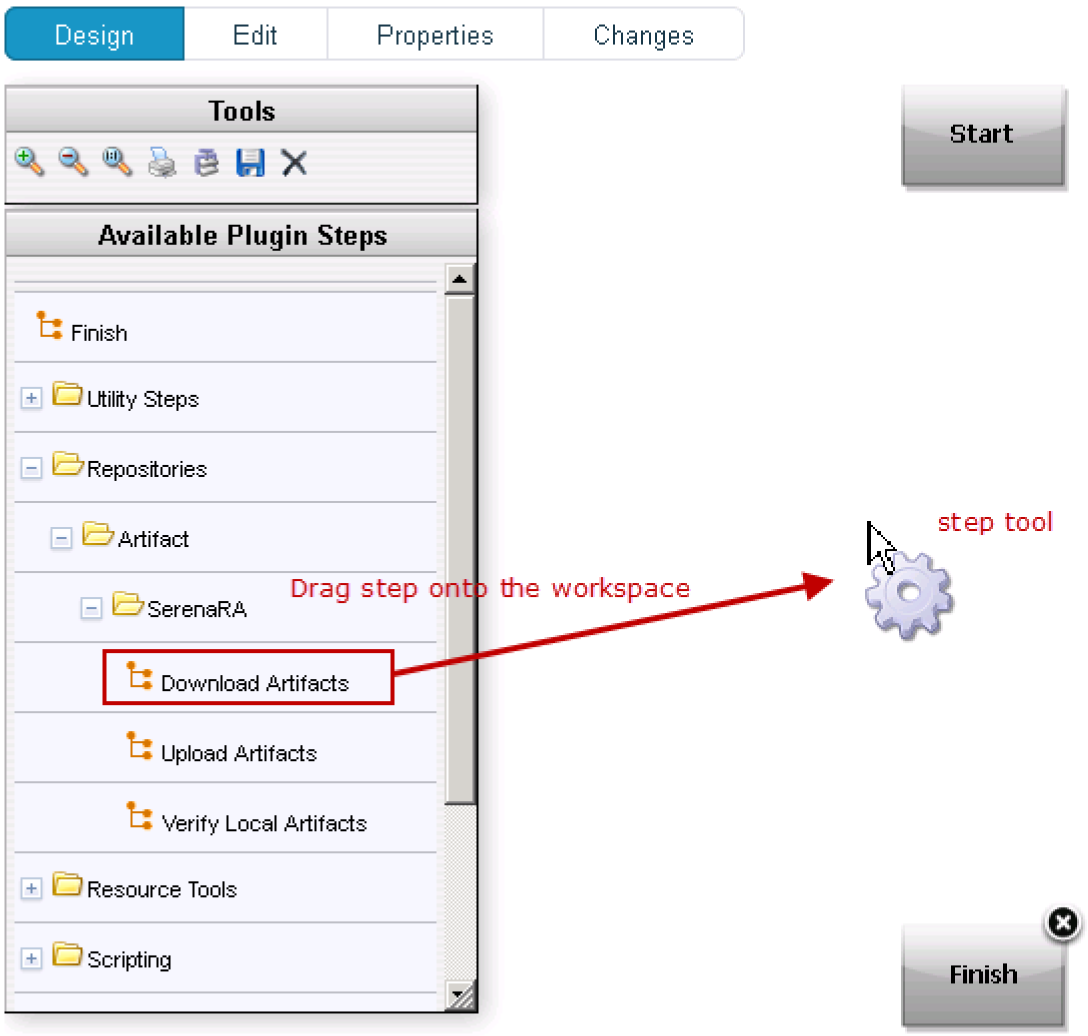
To add a step:

1. In the **Available Plug-in Steps** list, using your mouse drag the step you want to use onto the design space.

   The cursor changes to the *step tool*.

2. Release the step tool over the design space.

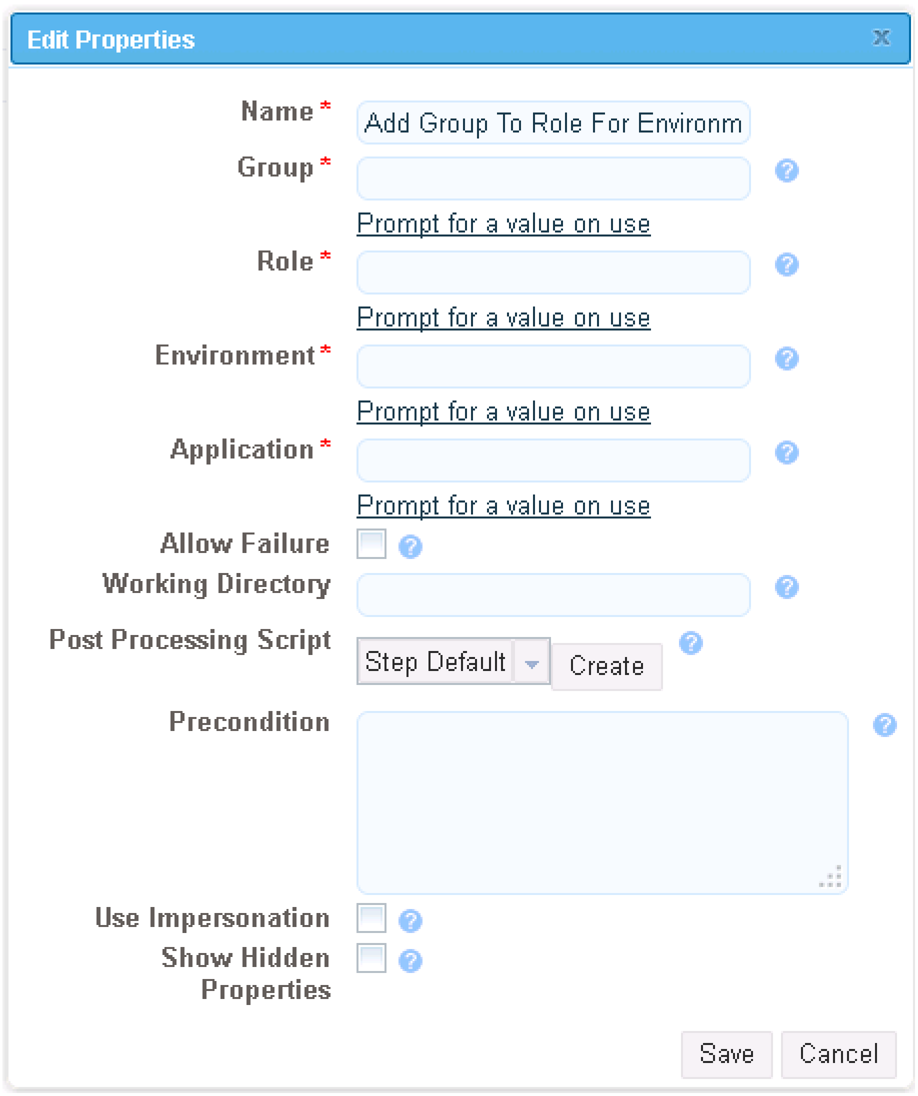
Figure 1. Adding a Step



The **Edit Properties** pop-up displays.

Because connections are created after configuring the step's properties, you can place the step anywhere on the design space. Steps can be dragged and positioned at any time.

Figure 2. Typical Edit Properties Pop-up



Edit Properties

Name * Add Group To Role For Environm

Group *

Prompt for a value on use

Role *

Prompt for a value on use

Environment *

Prompt for a value on use

Application *

Prompt for a value on use

Allow Failure

Working Directory

Post Processing Script Step Default Create

Precondition

Use Impersonation

Show Hidden Properties

Configuration dialogs are tailored to the selected step; only parameters associated with the step type are displayed.

3. After configuring the step's properties, to save the step click **Save**.

   The step is on the design space and ready to be connected to other steps.

4. If you want to remove the step from the design space, click **Cancel**.

   You can add connections immediately after placing a step, or place multiple steps before defining connections.

Connecting Process Steps

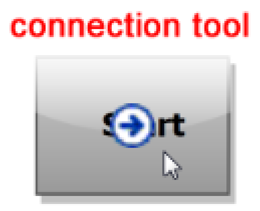
Connections control process flow. The originating step will process before the target step. Creating a connection between steps is a simple process; drag a connection from the originating step to the target step. Connections are formed one at a time between two steps, the originating step and the target step.

To create a connection:

1. Hover your mouse pointer over the step that you want to use as the connection's origin.

   The pointer changes to the connection tool.

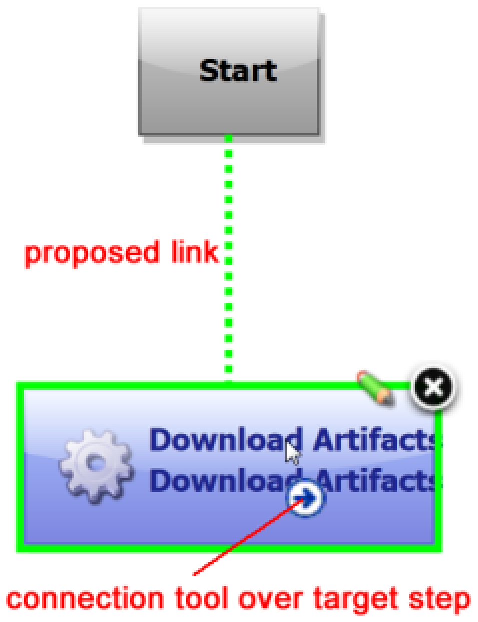
   Connection Tool

2. Drag the connection tool over the target step.

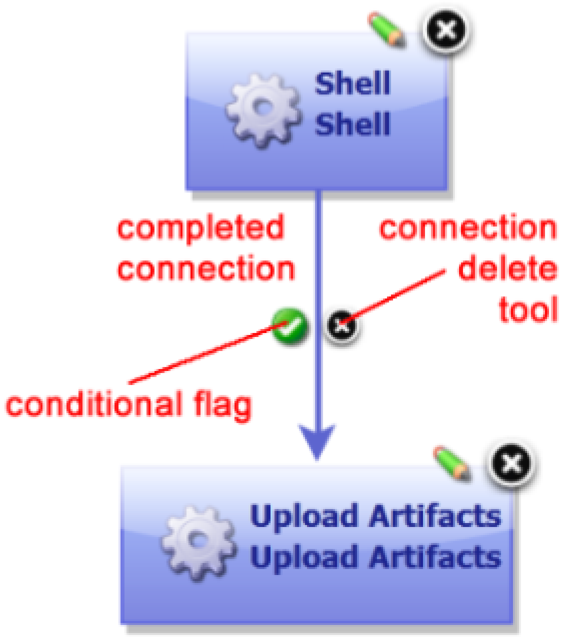
   When the target step is highlighted, release the mouse to create a connection.

   Dragging the Connection Over a Target Step

3. Release the connection tool over the target step to complete the connection.

   Completed Connection



Each connection has a connection delete tool, *conditional flag*, and might have others depending on the originating step. Remove a connection by clicking on the delete tool.

Process Properties

A *processing property* is a way to add user-supplied information to a process. A running process can prompt users for information and then incorporate it into the process. Properties are defined with the **Add Property** dialog.

To define a property:

1. On the **Properties** tab, click the **Add Property** button.
2. In the **Add Property** dialog, enter a name in the **Name** field.
3. Optional. Enter a description in the **Description** field.
4. Enter a label in the **Label** field.

   The label will be associated with the property in the user interface.
5. If the property is required, check the **Required** check box.

   Default value is unchecked–not required.
6. Specify the type of expected value with the **Type** drop-down list box.

   Supported types are: `text`, `text area`, `check box`, `select`, `multi select`, and `secure`. Default type is `text`.
7. In the **Default Value** field, enter a default value (if any).
8. To save your work, click the **Save** button. To discard changes, use the **Cancel** button.

To use a property in a process, reference it when you configure a step that uses it.

Switch Steps and Conditional Processes

Every connection (except connections from the Start step) has a delete tool and conditional flag. The conditional flag enables you to set a condition on a connection. The condition refers to the processing status of the originating step–success or failure. Possible flag conditions are:

success

The process completed successfully.

fail

The process did not finish successfully.

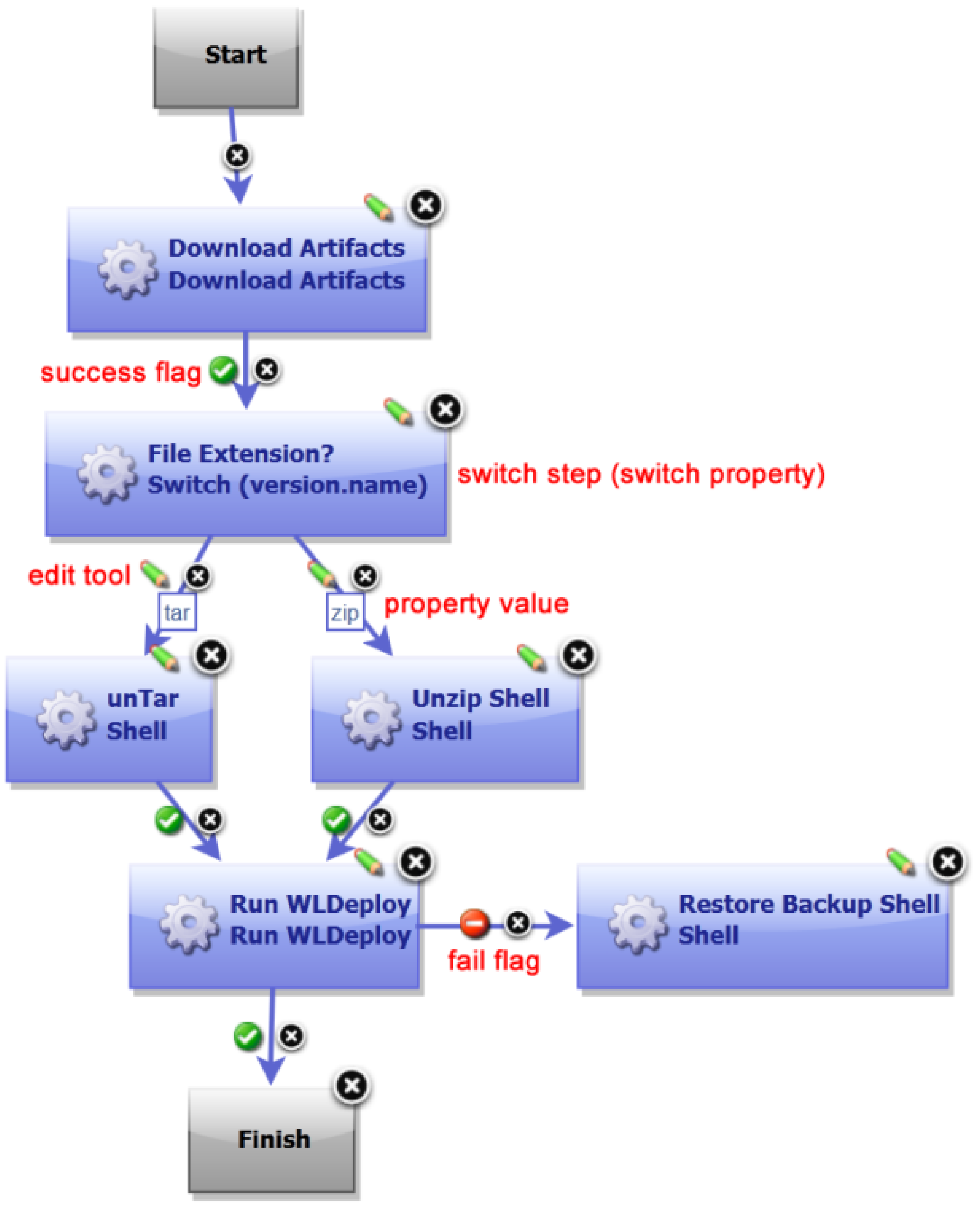
both

Accept either status.

By default, all connections have the flag set to checked (true), meaning the originating step must successfully end processing before the target step starts processing.

To change a flag's value, cycle through possible values by clicking the flag.

Process with Switch Step



A *switch step* is a utility step supplied by Serena Release Automation that enables process branching based on the value of a property set on the step. The accompanying figure illustrates a switch step. In this case, the switch property is `version.name`. The connections from the switch step represent process branches dependent on the value of `version.name`. In this example, regardless of which branch is taken, the process will proceed to the `Run WLDeploy` step.

Note: `Run WLDeploy` has success and fail conditions. See the *Serena Release Automation Plug-in Guide* for more information.

Note: If a step has multiple connections that eventually reach the same target step, determining whether the target will execute depends on the value of the intervening flags. If all of the intervening connections have success flags, the target will only process if all the steps are successful. If the intervening connections consist of an assortment of success and fail flags, the target will process the first time one of these connections is used.

For a process to succeed, execution must reach a Finish step. If it does not end with Finish, the process will fail every time.

Configuring Access for Process Steps

Serena Release Automation agents employ user impersonation when required to perform tasks for which they would not otherwise have permission. To run a database update script, for example, an agent might need to be the "oracle" user; but to update the application, the agent might need to be the "websphere" user.

By using impersonation, the same agent can run the script and update the application, which enables you to combine these steps into a single process. For information about user impersonation, see the User Impersonation [page 145] section.

User Impersonation

Serena Release Automation can use user impersonation when an agent must execute a command for which it might not otherwise have permission, or when a specific user must be employed for a given process. You implement impersonation when you configure a component's plug-in process step.

- On UNIX/Linux systems, the *su/sudo* commands are used to impersonate users, see User Impersonation on UNIX/Linux Using sudo (or su) [page 145].
- On Windows, Serena Release Automation provides a utility program to handle impersonation, see Impersonation on Windows Systems [page 146].

User Impersonation on UNIX/Linux Using sudo (or su)

For agents running on UNIX/Linux platforms, when you configure a process step you can provide the agent(s) with the user impersonation capability. When a process step that is impersonation-configured runs, the `sudo` (or `su`) command runs the step as the impersonated user.

Process steps can be considered individual shells; the `sudo` (or `su`) command enables a user to start a shell as another user.

Note: Each step that needs user impersonation must be configured independently. For more information about creating process steps, see Process Editor [page 135].

To configure impersonation using sudo:

1. Supply the username required by the target host.

2. Before `sudo` can be used:

   a. Password Required. Impersonation privileges *must* be defined in the `/etc/sudoers` file along with grant priviledges to run scripts from the agent .temp folder. For example:

      ```
      User1<>ALL=(User2)/home/User1/agent/var/temp/*
      ```

      Grants *User1* the rights to impersonate *User2* to run plug-in steps as *User2*.

      ```
      Defaults:X!requiretty
       X ALL=(Y)
      ```

      where *X* and *Y* are user names, and user *X* can run any command as user *Y*.

   b. No Password Required. Impersonation privileges *must* be defined in the `/etc/sudoers` file. For example:

      ```
      Defaults:X!requiretty
      X ALL=(Y) NOPASSWD: ALL
      ```

      where *X* and *Y* are user names, and user *X* can run any command as user *Y* without supplying a password.

When you create a process step, the `sudo` option is available for you to select. The `sudo` option is activated by default for plug-in steps. When the `sudo` check box is unchecked, the `su` option is active. However, the `su` option only applies to agents starting as root. Otherwise, `su` has no effect on the step or process. `su` *can* be used without configuring the `sudoers` file.

`su` and `sudo` maintain a record in the system logs of all of their activity.

For more information about `su/sudo` see the UNIX/Linux documentation.

Impersonation on Windows Systems

For agents running on Windows platforms, Serena Release Automation provides a program that handles impersonation.

You implement impersonation for Windows-based agents the same way you do for UNIX- or Linux-based agents:

When you configure a process step, you specify the *local user* credentials—user name and password—that will be used when the step is processed.

For impersonation purposes, a local user is:

- one whose user name and password are stored on the target computer
- who is part of the administration group
- has, at a minimum, the following privileges:

```
SE_INCREASE_QUOTA_NAME (adjust memory quotas for a process)
SE_ASSIGNPRIMARYTOKEN_NAME (replace a process-level token)
```

```
SE_INTERACTIVE_LOGON_NAME (local logon)
```

Impersonating the LocalSystem Account

You can also impersonate the Windows LocalSystem account. The LocalSystem account is installed on every Windows machine and is the equivalent of the root user on UNIX/Linux. It is guaranteed to have the privileges listed above.

Note: For Windows-based agents the `sudo` option is ignored if selected.

Component Manual Tasks

A component manual task is a mechanism used to interrupt a component process until some manual intervention is performed. A task-interrupted process will remain suspended until the targeted user or users respond. Typically, manual tasks are removed after the process has been tested or automated.

Similar to approvals, triggered tasks alert targeted users. Alerts are associated with component-, environment-, or resource-defined user roles. Affected users can respond—approve—by using the Work Items pane (see Work Items [page 168]). Unlike approvals, manual tasks are incorporated within a component process.

Creating Component Manual Tasks

To create a task:

1. Display the Create Task Definition dialog `Components > [`*`selected component`*`] > Tasks > Create Task Definition [`*`button`*`]`.

2. Name the task then select a template from the Template Name field.

   The individual tasks map to the notification scheme used by the application (see Creating Notification Templates [page 112]). If a scheme is not specified, the default scheme is used. The available notification schemes are:

   - ApplicationDeploymentFailure
   - ApprovalCreated
   - TaskCreated
   - ProcessRequestStarted
   - DeploymentReadied
   - ApplicationDeploymentSuccess
   - Approval Failed

Using Component Manual Tasks

Component manual tasks are implemented with the Manual Task component process step. Use the step to insert a manual task trigger into a component process.

Component Manual Task Properties

Field	Description
Name	Typically the name and description correspond to the component process.
Task Definition	Used to select a user-defined task, as described above.
Component Role	Select the role expected to respond. The user mapped to this role will have to respond to the generated work item before the process can continue.
Environment Role	Select the role expected to respond. The user mapped to this role will have to respond to the generated work item before the process can continue.
Resource Role	Select the role expected to respond. The user mapped to this role will have to respond to the generated work item before the process can continue.

If multiple roles are selected, all affected users will have to respond before the process can continue. For information about notification schemes, see Creating Notification Templates [page 112], and for information about creating component proceses see Process Editor [page 135].

Post-Processes

When a plug-in step finishes processing, its default post-processing element is executed. The post-processing element is defined in the plug-in's XML definition.

You can override the default behavior by entering your own script into the step's Post Processing Script field. A post-processing script can contain any valid JavaScript code. Although not required, it's recommended that scripts be wrapped in a `CDATA` element.

See the *Serena Release Automation Plug-in Guide* for more information.

Component Templates

There are two types of templates available:

component template

Enables you save and reuse component processes and properties and create new components from them; template-based components inherit the template properties and process(es).

configuration template

Typically used to save server or property configurations.

Creating a Component Template

To create a template:

1. Display the Create Component Template dialog `Components > Templates > Create Template [`*`button`*`]`.
2. Enter the template's name in the **Name** field.
3. Enter a description in the **Description** field.

   The optional description can be used to convey additional information about the template.
4. Select a plug-in from the **Status Plug-in** field.

   If you previously created any status-related plug-ins, they will be listed here. The default value is `Default`, meaning that the template will have Serena Release Automation-supplied steps available for use.
5. Select the source for the artifacts from the **Source Config Type** drop-down list.

   Selecting a value other than the default `None`, displays additional fields associated with your selection. Source-dependent fields are used to identify and configure the artifacts. If you select a source, components based on the template will use the same source (see Basic Fields [page 122]).

   Note: If you select a source, any properties you configure will be set for any components created with the template.
6. Click the **Save** button to save the template.

   Saved templates are listed in the **Component Templates** pane.

You create a process for the template in the same way processes are created for components. For information about creating component processes, see Component Processes [page 134].

Importing/Exporting Templates

Templates can be imported and exported.

Exporting Templates

Exporting a template creates a JSON file (file extension `json`) that contains the template's configuration information, properties, and processes.

To export a template:

- On the Component Templates pane `Components > Templates`, click the `Export` link in the Actions field. You can load the file into a text editor, or save it. If you save it, a file is created with the same name as the selected component, for example, `helloWorldTemplate.json`.

Importing Templates

When you import a template, you can create an entirely new template or upgrade an existing one.

To import a template:

1. Display the Import Template dialog `Components > Templates > Import Template [button])`.
2. Enter the path to the JSON file containing the template or use the Browse button to select one.
3. If you want to upgrade an existing template, check the Upgrade Template check box. To create a new template, leave the box unchecked.

   If the template's name in the JSON file (not the name of the file itself) matches an existing template, the template will be upgraded. If the name is not found, the command has no effect.

   Note: The template's name is the first parameter in the JSON file; for example,

   ```
   "name": "helloWorldTemplate",
   ```
4. Click Submit.

Component Template Properties

Component template properties ensure that every component created from a template has the same properties. The three types of available properties are described in the following table.

Component Template Properties table

Template Property Type	Description
Properties	Custom property. Every component will inherit the value defined in the template (it cannot be overridden by a component). If you change the value, the change will be reflected in components created from the template, including those previously created.

Template Property Type	Description
Component Property Definitions	Every component will have this property; it will appear on the Create Component dialog for every component created from this template (see Creating Components [page 122]). A value defined here can be changed by created components. Each property must have a type: • `Text` Enables users to enter text characters. • `Text Area` Enables users to enter an arbitrary amount of text, limited to limited to 4064 characters. • `Check Box` Displays a check box. If checked, a value of *true* will be used; otherwise the property is not set. • `Select` Requires a list of one or more values which will be displayed in a drop-down list box. Enables a single selection. **NOTES** Not currently implemented. • `Multi Select` Requires a list of one or more values which will be displayed in a drop-down list box. Enables multiple selections. • `Secure` Used for passwords. Similar to Text except values are redacted.

Template Property Type	Description
Environment Property Definitions	Every environment that uses a component created by this template will have this property. The property will appear on the environment's Component Mappings pane `Applications > [`*`selected application`*`] > Environments > [`*`selected environment`*`] > Component Mappings`), see Application Environments [page 160]. A value defined here can be changed by environment. Each property must have a type: • `Text` Enables users to enter text characters. • `Text Area` Enables users to enter an arbitrary amount of text, limited to limited to 4064 characters. • `Check Box` Displays a check box. If checked, a value of `true` will be used; otherwise the property is not set. • `Select` Requires a list of one or more values which will be displayed in a drop-down list box. Enables a single selection. **NOTES** Not currently implemented. • `Multi Select` Requires a list of one or more values which will be displayed in a drop-down list box. Enables multiple selections. • `Secure` Used for passwords. Similar to Text except values are redacted.

Using Component Templates

When you create a component based on a template, the component inherits any process(es) the template may have (see Component Processes [page 134]), and any possible properties (see Component Properties [page 128]).

To create a template-based component:

1. Display the Create Component dialog `Components > Templates > [`*`selected template`*`] > Create Component [`*`button`*`]`.

   The Create Component dialog (the same dialog used to create non template-based components) is used to configure component. Properties defined in the template will be predefined. If a source was selected in the template, the source is set here and the Source Config Type field is locked. For information about using this dialog, see Creating Components [page 122].

2. After configuring editable properties, save the component.

Templates used to create components are listed in the Templates view.

Components created from templates are listed in the Components view.

Configuration Templates

Configuration templates contain configuration data. Typically, the data is for server configurations, for example on Tomcat servers, but the data can be for any purpose.

To create a configuration template:

1. Display the Create Configuration Template dialog `Components > [`*`selected component`*`] > Templates > Create New Configuration Template [`*`button`*`]`.
2. Enter a name in the Name field.
3. In the Template field, enter or paste the template text. Text can be in any script, or no script at all. The amount of text is based on the database used by Serena Release Automation. In general, there is no limit to the amount of text used for a configuration template.
4. Save your work when you are finished.

Configuration templates can be edited at any time by using the Edit action.

Deleting and Deactivating Components

Components can be deactivated or deleted. To delete or deactivate a component, use the desired action on the Components pane for the intended component.

When a component is deactivated, it remains in the database and CodeStation and can be activated later. To activate a component, first click the Show Inactive Components check box, then use the Activate action for the component.

When a component is deleted, it is removed–along with all versions–from the database and CodeStation and cannot be activated at a later time. The original artifacts are not affected; only the CodeStation copies are deleted.

Note: Components cannot be deleted if they are used by an application. To delete a component used by an application, first remove the component from the application.

Chapter 15: Managing Applications

Applications are responsible for bringing together all the components that need to be deployed together. This is done by defining the different versions of each component as well as defining the different environments the components must go through on the way to production. In addition, applications also map the constituent hosts and machines (called resources) a component needs within every environment.

Applications also implement processes such as automated deployments and rollbacks. These are called *application processes*. The main difference between application-level processes and component-level processes is that the former are typically concerned with deployment and the latter are concerned with running commands.

Applications also introduce snapshots to manage the different versions of each component. A *snapshot* represents the current state of an application in the environment. Typically, the snapshot is generated in an environment that has no approval gates – called an *uncontrolled environment*. For most users, the snapshot is pushed through the pipeline.

Note: Before configuring an application, you will need to ensure that at least one agent has been installed in a target environment. In addition, you will also need to add at least one resource group to the agent (see Chapter 16: Managing Resources [page 173]).

Related Topics

- Chapter 16: Managing Resources [page 173]
- Chapter 14: Managing Components [page 121]
- Chapter 17: Configuration and Processes Options [page 179]

Application Creation Overview

Applications associate components with the agents that will manage them, and define processes to perform deployments.

The following table summarizes the steps performed to create applications.

Application Creation Steps table

Step	Description
1. Create an application and identify its components.	After defining the application, identify the components it will manage. Associating a component makes its processes and properties available to the application. An application can have any number of components associated with it.

Step	Description
2. Create an environment	Define an environment and use it to map an agent to component(s). Mapping means assigning an agent to manage the component. Each component can be mapped to the same agent, a different one, or some combination. An application can have more than one environment defined for it.
3. Create an application process	Use the process design editor to create a process. Application processes are created with the same editor used to create the component process, but uses a different toolkit of process steps. Previously defined component processes can be incorporated into the process.

Related topics:

- Chapter 21: Serena Release Automation Properties [page 219]
- Snapshots [page 168]
- Importing/Exporting Applications [page 158]

Creating Applications

You can create an application from scratch or import an existing one (see Importing/ Exporting Applications [page 158] for information about importing applications.

After creating an application, you:

- add components (Adding Components to an Application [page 157])
- create an environment (Creating an Environment [page 160])
- associate an agent with the environment (Mapping Resources to an Environment)
- create an application process (Application Processes [page 162])

Before configuring an application, ensure that at least one agent has been installed in a target environment. See Chapter 16: Managing Resources [page 173].

To create an application:

1. Display the Create Application dialog `Applications > Create Application [button]`, and enter the following:

   New Application Information table

Field	Description
Name and Description	Typically, correspond to the application you plan on deploying.

Field	Description
Notification Scheme	notifications—based on events—can be sent out due to Serena Release Automation integrations with LDAP and e-mail servers. For example, when an application deployment fails or succeeds, the default notification scheme sends out an email. Notifications can also be used to send out emails to a user or a group (based on their security role) for approval of a requested deployment (see Creating Notification Templates [page 112]).
Enforce Complete Snapshots	If selected, the application requires every component to get versioned.

2. Save your work when done.

Adding Components to an Application

Add at least one component to the application. Applications bring the different components (their versions and processes) together so they can be deployed as a single unit.

To add components to an application:

1. Display the Add a Component dialog `Applications > [`*`select application`*`] > Components > Add Component [`*`button`*`]`.
2. Use the Select a Component list box to select one component at a time.

Adding Application Configuration Properties

To add a property to the selected application:

1. Use the `Add Property` button.

   The Edit Property pop-up displays.
2. Enter the property's name in the Name field.

   While component fields can be of any size, configuration properties are restricted to 4,000 characters.
3. Enter a description of the property in the Description field.
4. Specify whether the property is secure by using the Secure check box.

   Secure properties are stored encrypted and displayed obscured in Serena Release Automation's user interface.
5. Enter a value for the property in the Value field.
6. To save the property, click Save, or to discard your work click Cancel.

Modifying and Deleting Application Configuration Properties

Modifying Application Configuration Properties

To modify a previously created property, use the `Edit` link in the Action column to display the Edit Property pop-up.

Deleting Application Configuration Properties

To delete a property, use the `Delete` link in the Action column.

Importing/Exporting Applications

Applications can be imported and exported. Importing/exporting can be especially useful if you have multiple Serena Release Automation servers, for example, and need to quickly move or update applications.

Related Topics

- Exporting Applications [page 158]
- Importing Applications [page 158]

Exporting Applications

Exporting an application creates a JSON file (file extension `json`) that contains the application's properties, components (and their associated properties and processes), and processes. For information about JSON, see http://www.json.org/.

To export an application:

On the Applications pane `Management > Applications`), Actions field, click the `Export` link. You can load the file into a text editor, or save it. If you save it, a file is created with the same name as the selected component, for example, `helloWorldApplication.json`.

Importing Applications

When you import an application, you can create an entirely new application or upgrade an existing one. Components—including their properties and processes—associated with the application are also imported (if available to the importing server). For information about templates associated with imported components, see Importing/Exporting Components [page 129].

Note: If imported components have the Import Versions Automatically parameter set to true, Serena Release Automation will automatically import component versions as long as the artifacts are accessible to the importing server.

To import an application:

1. Display the Import Application dialog (`Applications > Import Application [`*`button`*`])`.
2. Enter the path to the JSON file containing the application definition or use the Browse button to select one.
3. If you want to upgrade an existing application, check the Upgrade Application check box. To create a new application, leave the box unchecked.

If the application's name in the JSON file (not the name of the file itself) matches an existing application, the application's parameters are updated with new values, and new items—such as processes, environments, and components—are added. If the name is not found, the command has no effect.

Note: The application's name is the first parameter in the JSON file. For example:

```
"name": "helloWorldApplication",
```

4. Specify how imported components should be handled with the Component Upgrade Type drop-down box. For these options, the components must be on the importing server.

   - To use the same components used by the imported application, select Use Existing Component. The new application will contain references to the imported applications components. This option is especially useful if you are importing a lot of applications.

     If you are upgrading, the application will use the imported components, and no longer use any not used by the imported application.

   - To create new components based on those used by the imported application, select Create Component. New components will be created (based on the imported application's components).

     If you are upgrading, the application will use the newly created components and no longer use any it previously used.

   - When you want to create a fresh installation, select Fail if Component Exists. If you are creating an application, it will create both a new application and component unless the component already exists, in which case the application is not imported.

     If you are upgrading, the upgrade will fail if any imported components already exist on the importing server.

   - To ensure a component is on the importing server, select Fail if Component Does Not Exist. If you are creating an application, it will create both a new application and component unless the component does not exist, in which case the application is not imported.

     If you are upgrading, the upgrade will fail if an imported component does not already exist on the importing server.

   - To upgrade existing components, select Upgrade if Exists. This option creates an application and upgrades existing components with data from the imported application.

     If you are upgrading and existing components match imported ones (all must match), the components will be upgraded. If none of the imported components match existing ones, the imported components will be used.

5. Click Submit.

Application Environments

An *environment* is a user-defined collection of resources that hosts an application. An environment is the application's mechanism for bringing together components with the agent that actually deploys them. Environments are typically modeled on some stage of the software project life cycle, such as development, QA, or production. A resource is a deployment target, such as a database or J2EE container. Resources are usually found on the same host where the agent that manages them is located. A host can be a physical machine, virtual machine, or be cloud-based.

Environments can have different topologies—for example: an environment can consist of a single machine; be spread over several machines; or spread over clusters of machines. Environments are application scoped. Although multi-tenant machines can be the target of multiple applications, experience has shown that most IT organizations use application-specific environments. Additionally, approvals are generally scoped to environments.

Serena Release Automation maintains an inventory of every artifact deployed to each environment and tracks the differences between them.

Creating an Environment

Before you can run a deployment, you must define at least one environment that associates components with an agent on the target host. This initial environment is typically uncontrolled and often used to create snapshots.

To create an environment:

1. Display the Create Environment dialog `Applications > [`*`select application`*`] > Environments > Add New Environment [`*`button`*`]`.

2. Enter the environment information in the fields provided:

   New Application Information table

Field	Description
Name and Description	Name is used as part of the deployment process, and typically corresponds to the target environment. For example, if you are deploying to an integration environment, you may want to use the name "SIT". Description is optional text.
Require Approvals	To require an approval before components can be deployed to the environment, select this check box. If checked, Serena Release Automation will enforce an approval process before the deployment can be deployed to the environment. Initial deployments are typically done in uncontrolled environments, but once the deployment is successful, you can configure an approvals process as the application moves along the development pipeline. If you are setting up more than one environment, consider creating an approvals process for at least one of them.
Lock Snapshots	If you want all snapshots used in this environment to be locked to prevent changes, select the check box.

Field	Description
Color	Select a color to visually identify the environment in the user interface.
Inherit Cleanup Settings	Determines how many and for how long componenet versions are kept in CodeStation. By default, this check box is selected; the application will use the values specified on the System Settings pane. If unchecked, additional fields display: the Days to Keep Versions (initially set to -1, keep indefinitely) and Number of Versions to Keep (initially set to -1, keep all), which enable you to define custom values.

3. Click Save.

Mapping Resources to an Environment

1. After you have added a component to the application, define where its artifacts should be deployed by selecting a resource (agent) or resource group. See Chapter 16: Managing Resources [page 173].
2. Display the Component Mappings pane `Applications > [`*`selected application`*`] > Environments > [`*`selected environment`*`] > Component Mappings`.
3. If the application has several components associated with it, select the one you want to use from the component list. Each component associated with this application can be mapped to a different agent (resource).
4. To associate a resource with the selected component:
   - To add a resource group, click the Add a Resource Group button and select a resource group. For information about creating resources, see Resource Groups [page 173].
   - To add a resource, click the Add a Resource button and select an resource.

After mapping components and resources, make the application deployment ready by creating an application process, which is described in the following section.

Environment Properties

Environment properties can be created with the environment's Properties pane (`Applications > [`*`selected application`*`] > Environments > [>`*`selected environment`*`] > Properties`.

A value set on component environment overrides one with the same name set directly on an environment property. Component environment properties enable you to centralize properties, tracking type and default values, for instance. Environment properties provide ad-hoc lists of `property=value pairs`.

Referenced: `${p:`*`environment/propertyName`*`}`

Application Processes

Application processes, like component and standalone processes, are created with the process editor. Serena Release Automation provides several common process steps, otherwise application processes are configured from processes defined for their associated components.

Application processes can run manually, automatically on some trigger condition, or on a user-defined schedule. When a component has several processes defined for it, the application determines which ones are executed and in which order.

An application process is always associated with a target environment. When an application process executes, it interacts with a specific environment. At least one environment must be associated with the application before the process can be executed.

Application processes are environment agnostic; processes can be designed independently of any particular environment. To use the same process with multiple environments, you associate each environment with the application and execute the process separately for each one.

In addition to deployments, several other common processes are available, such as rolling back deployments. Serena Release Automation tracks the history of each component version, which enables application processes to restore environments to any desired point.

Creating Application Processes

To create an application process:

1. Display the Create an Application Process dialog `Applications > [`*`select application`*`] > Create New Process [`*`button`*`]`.

2. Enter the following information:

   Application Process Fields table

Field	Description
Name/ Description	Typically the name and description correspond to the application you plan on deploying.
Required Application Role	Use this drop-down to select the role a user must have in order to run the application. For information about creating application roles, see Roles and Permissions [page 94]. The default value is `None`.
Inventory Management	If you want to handle inventory manually, select `Advanced`. To have inventory handled automatically, leave the default value, `Automatic`, selected.

Field	Description
Offline Agent Handling	Specify how the process reacts if expected agents are offline: **Check Before Execution** Checks to see if expected agents are on line before running the process. If agents are off line, the process will not run. **Use All Available; Report Failure:** Process will run as long as st least one agent defined in the environment is on line; reports any failed deployments due to off line agents. Useful for rollbacks or configuration deployments. **Always Report Success** Process will run as long as at least one agent defined in the environment is on line; reports successful deployments.

3. Save your work.

Application Process Step Details

After you create an application process, select it to design the process steps. The application process steps are described in the following topics:

- Finish [page 163]
- Install Component [page 163]
- Uninstall Component [page 164]
- Rollback Component [page 165]
- Manual Application Task (Utility) [page 166]

Finish

Ends processing. A process can have more than one Finish step.

Install Component

Installs the selected component using one of the processes defined for the component.

Install Component Properties table

Field	Description
Name	Can be referenced by other process steps.

Field	Description
Component	Component used by the step; a step can affect a single component. All components associated with the application are available. If you want to install another component, add another install step to the process.
Use Versions Without Status	Restricts the components that can be used by the step—components with the selected status are ignored. Available statuses: `Active` means ignore components currently deployed; `Staged` means ignore components currently in pre-deployment locations.
Component Process	Select a process for the component selected above. All processes defined for the component are available. Only one process can be selected per step.
Ignore Failure	When selected, the step will be considered to have run successfully.
Limit to Resource Role	User-defined resource role the agent running the step must have.
Run on First Online Resource Only	Instead of being run by all agents mapped to the application, the step will only be run by the first online agent identified by Serena Release Automation. The mechanism used to identify the "first" agent is database-dependent (thus indeterminate).
Precondition	A JavaScript script that defines a condition that must exist before the step can run. The condition must resolve to true or false.

Uninstall Component

Uninstalls the selected component.

Uninstall Component Properties table

Field	Description
Name	Can be referenced by other process steps.
Component	Component used by the step; a step can affect a single component. All components associated with the application are available. If you want to install another component, add another install step to the process.
Remove Versions With Status	Restricts the components that are affected by the step, only components with the selected status are affected. Available statuses: `Active` means use components currently deployed; `Staged` means use components currently in pre-deployment locations.

Field	Description
Component Process	Select a process for the component selected above. All processes defined for the component are available. Only one process can be selected per step.
Ignore Failure	When selected, the step will be considered to have run successfully.
Limit to Resource Role	User-defined resource role the agent running the step must have.
Run on First Online Resource Only	Instead of being run by all agents mapped to the application, the step will only be run by the first online agent identified by Serena Release Automation. The mechanism used to identify the "first" agent is database-dependent (thus indeterminate).
Precondition	A JavaScript script that defines a condition that must exist before the step can run. The condition must resolve to true or false.

Rollback Component

Rolls back a component version; replaces a component version with an earlier one.

Rollback Component Properties

Field	Description
Name	Can be referenced by other process steps.
Component	Component used by the step; a step can affect a single component. All components associated with the application are available. If you want to install another component, add another install step to the process.
Remove Versions With Status	Restricts the components that are affected by the step, only components with the selected status are affected. Available statuses: `Active` means use components currently deployed; `Staged` means use components currently in pre-deployment locations.
Component Process	Select a process for the component selected above. All processes defined for the component are available. Only one process can be selected per step.
Ignore Failure	When selected, the step will be considered to have run successfully.
Limit to Resource Role	User-defined resource role the agent running the step must have.

Field	Description
Rollback type	Determines the type of rollback. Available statuses: `Remove Undesired Incremental Versions` and `Replace with Last Deployed`.
Run on First Online Resource Only	Instead of being run by all agents mapped to the application, the step will only be run by the first online agent identified by Serena Release Automation. The mechanism used to identify the "first" agent is database-dependent (thus indeterminate).
Precondition	A JavaScript script that defines a condition that must exist before the step can run. The condition must resolve to true or false.

Manual Application Task (Utility)

A manual task is a mechanism used to interrupt an application process until some manual intervention is performed. A task-interrupted process will remain suspended until the targeted user or users respond. Typically, manual tasks are removed after the process has been tested or automated.

Similar to approvals, triggered tasks alert targeted users. Alerts are associated with environment- or application-defined user roles. Affected users can respond—approve—by using the Work Items pane (see Work Items [page 168]). Unlike approvals, manual tasks can be incorporated within an application process.

The task used to configure this step must have been previously defined with the Application Manual Tasks [page 167].

Manual Application Task Properties table

Field	Description
Name	Typically the name and description correspond to the application.
Task Definition	Used to select a user-defined task.
Environment Role	Select the role expected to respond. The user mapped to this role will have to respond to the generated work item before the process can continue.
Application Role	Select the role expected to respond. The user mapped to this role will have to respond to the generated work item before the process can continue.

If both roles are selected, all affected users will have to respond before the process can continue. See Creating Notification Templates [page 112].

Application Manual Tasks

A manual task is a mechanism used to interrupt an application process until some manual intervention is performed. A task-interrupted process will remain suspended until the targeted user or users respond. Typically, manual tasks are removed after the process has been tested or automated.

Similar to approvals, triggered tasks alert targeted users. Alerts are associated with environment- or application-defined user roles. Affected users can respond—approve—by using the Work Items pane (see Work Items [page 168]). Unlike approvals, manual tasks can be incorporated within an application process.

Creating Application Manual Tasks

To create a task:

1. Display the Create New Task Definition dialog `Applications > [`*`selected application`*`] > Tasks > Create Task Definition [`*`button`*`]`.
2. Name the task then select a template from the Template Name field.

   The individual tasks map to the notification scheme used by the application(see Creating Notification Templates [page 112]). If a scheme is not specified, the default scheme is used. The available tasks are:

   - ApplicationDeploymentFailure
   - ApprovalCreated
   - TaskCreated
   - ProcessRequestStarted
   - DeploymentReadied
   - ApplicationDeploymentSuccess
   - Approval Failed

Using Manual Tasks

Manual tasks are implemented with the Manual Application Task (Utility) [page 166]. Use the step to insert a manual task trigger into an application process.

Approval Process

An approval process enables you to define the job that needs approved and the role of the approver. An approval process must be created if the Requires Approval check box is selected when creating/editing an environment. If a scheduled deployment requiring approval reaches its start time without approval given, the process will not run and act as a rejected request. To resubmit a request, you must request a new process. If an approval-requesting process does not have a scheduled deployment time, the process will remain idle until a response has been made.

Creating an Approval Process

To create an approval process, display the Approval Process Design Pane.

`Management > Applications > [`*`Application_Name`*`] > Environments > Environment: [`*`Environment_Name`*`>] Approval Process`

Once the pane is displayed, select the steps that need approval from the process editor. The steps are based on job type and the role of the approver. You have the option of selecting three job types: the Application, Component, and/or Environment. For help using the process editor see Process Editor [page 135].

Reviewing Status

To view the status of the request, display the Deployment Detail pane on the Reports page. If a request has been approved it will display as success. However, if the request was rejected it will show failed. If a request is failed display the Application Process Request by clicking view request.

If a comment has been made regarding the process, you can view it by clicking the log button in the actions column on the Application Process Request.

Work Items

If a job requiring approval is created, an approval process will have to be created. The job requiring approval will display in the approvers Work Items page. Until approved, the job will remain idle if unscheduled. If time has elapsed on a scheduled job needing approval, the job will fail. This control allows the approver to verify the deployment details, and choose the time it is deployed. Notifications are sent to users who are eligible to complete an approval step if the system is configured with an email server and the user has an email address set.

View Details of Process

In the Works Items page, the approver can view the name of the process, when the request was submitted, who requested the process, and the snapshot or version used. The approver can also view details of the environment or resource by clicking the link in the Environment/Resource column. They can view the details of the target by clicking the link in the target column. Or view details on the request by selecting the View Request in the Actions column. The Actions column is also where the approver can respond to the request.

Responding to Request

To respond to a request, display the Respond dialog box by clicking Respond in the Actions column. The approver has the option of leaving a comment. If a request is rejected the process will not run. If approved, the process will begin.

Snapshots

Snapshots specify what combination of component versions you deploy together. They are models you create before deploying the application.

A snapshot specifies the exact version for each component in the application. When a snapshot is created, Serena Release Automation gathers together information about the application, including the component versions, for a given environment.

Typically, the snapshot is generated in an environment that has no approval gates, which is called an uncontrolled environment. For most users, the snapshot is pushed through the pipeline.

Typically, one of the environments will always remain uncontrolled to allow for snapshots. When a successful deployment has been run in the uncontrolled environment, a snapshot is created based on the application's state within the environment: thus capturing the different versions of the components at that time.

As the application moves through various testing environments, for example, Serena Release Automation ensures that the exact versions (bit for bit) are used in every environment. Once all the appropriate stages and approvals for a snapshot are complete, the snapshot is pushed to production.

Creating Snapshots

To create a snapshot:

1. Display the New Application Snapshot pop-up `Management > Applications > [select application] > Snapshots [tab] > Create Snapshot [button].`
2. Enter the name and description of your snapshot in the Name and Description fields.
3. Click Save.

Snapshot Options

Options available for snapshots are given in the following table:

Option	Description
Dashboard tab	
Add a Status button	Assign a status to the selected snapshot.
Request Process button	Run the application process.
Preview button	Preview the application process.
Component Versions tab	
Add Versions link	Enables you to select any version in Codestation for the component.
Copy from Environment button	Uses the currently deployed (in this environment) component version.
Copy from Snapshot button	Uses the currently deployed (in this environment) component version. **Warning:** This action will remove any existing configuration from the snapshot.

Option	Description
Configuration tab	
Reset All to Latest	This action will reset all configuration targets to the latest version.
Lock Loose Entries	This action will lock all configuration targets that are automatically using the latest version available. Are you sure you want to lock all loose configuration targets?
Copy from Snapshot	Uses the currently deployed (in this environment) component configuration. **Warning:** This action will remove any existing configuration from the snapshot.
Edit tab	
Name	Modify the name of your snapshot .
Description	Modify the description of your snapshot.
Calendar tab	
View Date field	Set the start date to view on the calendar.
Month / Week / 3 Days / 1 Day links	Set the time period to view of the calendar.
Schedule Process button	Schedule the application process

Application Gates

Gates provide a mechanism to ensure that component versions cannot be deployed into environments unless they have the gate-specified status. Version statuses are user-defined values that can be applied to component versions and used in component processes or application gates. Version statuses can be applied though the user interface `Management > Components > [`*`selected component`*`] > Versions > [`*`selected version`*`] > Add a Status [`*`button`*`]`, or by the Add Status to Version plug-in step. They are displayed in the Latest Status field on the component's Versions pane `Management > Components > [`*`selected component`*`] > Versions`.

Version statuses are defined in the `default.xml` file which you can freely edit to add your own values, see Structure of the default.xml File [page 171]. Component versions do not have to have gates. Gates are defined at the environment level; an environment can have a single gate defined for it.

Creating Gates

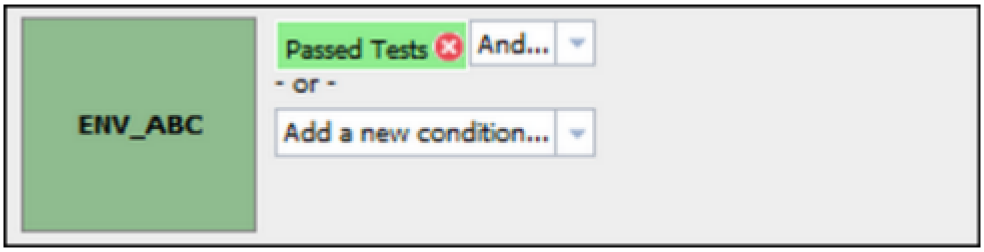
To create a gate:

1. Display the Gates pane for the target application `Applications > [`*`selected application`*`] > Gates`.
2. Select a value from the Add a new condition list box.

   The available statuses are defined in the `default.xml` file (discussed below). The default statuses— `Latest`, `Passed Tests`, `Archived`—are supplied as examples; it is assumed you will supply your own values.

   Selecting a value provides both `And` and `Or` selection boxes.

   Figure 1. Gate Definition

   Using the `And` box adds an additional value to the condition that must be satisfied. Using the default values for example, defining the following gate `Passed Tests And Latest` means that only component versions with both statuses— `Passed Tests` and `Latest`—satisfy the condition and can be deployed into the environment. A single condition can have as many `And`-ed values as there are statuses defined in the `default.xml` file.

   Using the `Or` box adds an additional condition to the gate. Additional conditions are defined in the same way as the first one. A gate with two or more conditions means the component will be deployed if it meets *any* of the conditions. For example, if the following two gates are defined, `Passed Tests`, and `Latest`, a component will pass the gate if it has either status (or both). A single gate can have any number of conditions.
3. Save your work when finished.

See Component Version Statuses [page 133] for more information about component statuses.

Structure of the default.xml File

Inventory and versions statuses are defined in the `default.xml` file. You can modify the supplied values for both types as well as add your own values. If you modify the file, restart the server in order to see your changes.

Here an example of `default.xml`:

```
<?xml version="1.0"?>
  <status-scheme name="Default">
  <inventory-statuses>
```

```
<status name="Active" color="#8DD889" unique="true" />
<status name="Staged" color="#80D8FF" />
</inventory-statuses>
<version-statuses>
<status name="Latest" color="#F6F4D8" unique="true" />
<status name="Passed Tests" componentRoleName="StatusAdder" color="#FFDDAA" />
<status name="Archived" color="#AAAAAA" />
</version-statuses>
</status-scheme>
```

Each `<status/>` element has a required `name` attribute and several optional ones, defined in the following table:

<status> Attributes table

Attribute	Description
name	Identifies the status; appears in user-interface. Used to create gates, and available in process steps.
color	Hexadecimal color definition; determines the color displayed in the user interface.
unique	Boolean value (true|false). Only one component version with this status/attribute will be deployed to the environment.
componentRoleName	Security role required by user to add this status to the component version.

All attributes must be enclosed in double-quotes.

Note: While you can add as many values as you like, you cannot create new status types (only inventory- and version-statuses are supported). Additionally, all values must be defined in `default.xml`.

Chapter 16: Managing Resources

To run a deployment, Serena Release Automation requires an agent (resource) or proxy agent on the target machine. Typically, an agent is installed in every environment that an application passes through.

A typical production pipeline might be, say, SIT, UAT, PROD (the application passes through two testing environments before reaching production). In this scenario, at least three agents need to be installed, one per environment. If different components run on different machines within a given environment, you may want to install multiple agents in that environment.

Whether you need one or multiple resources per environment is determined by your current infrastructure, deployment procedures, and other requirements. Many Serena Release Automation users have significant differences among environments; in SIT you might need to deploy a component to one machine, while in UAT you might need to deploy the component to multiple machines.

For example, you could configure sub-groups for the single agent in the SIT environment and then set up individual resources for each agent in the UAT environment.

Resource Groups

Serena Release Automation uses the concept of resource groups to help you organize and manage the agents installed in different environment throughout the network.

You need to create at least one resource group per installed agent, as when configuring your Processes you will need to select the appropriate Group. What groups you create and how you organize the groups, such as using subgroups, depends on your existing organizational processes and infrastructure.

Note: Before continuing, ensure that at least one agent has been installed in a target environment.

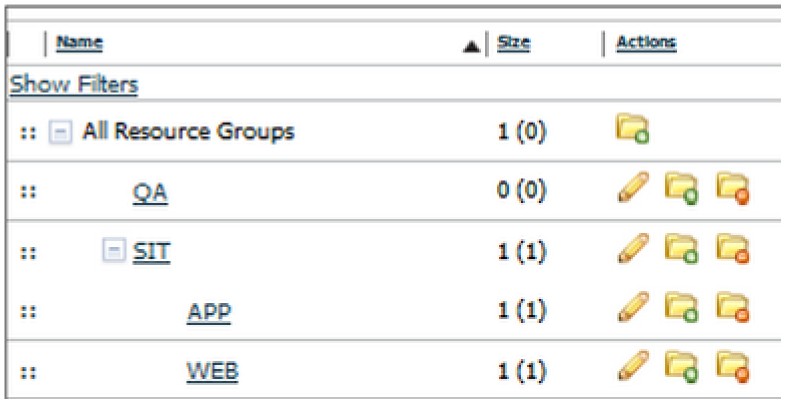
Creating a Resource Group

To create a resource group:

1. Go to `Management > Resources > Groups` and click the folder icon.
2. Enter Name and description. Typically, the name will correspond to either the Environment the Resource participates in, the Component that uses the Resource Group, or a combination of both, for example SIT, DB, or SIT-DB. Give a description that tells how you intend to use the Resource to which this Group is assigned.
3. Select the Type. Typically Static is used.
4. Once the Resource has been created, select the pencil icon to edit the Group.
5. Once you assign a Group to a Resource, you add Subresources. A *subresource* enables you to apply logical identifiers, or categories, within any given Group.

During deployment configuration, you can select a given Subresource that the Process will run on. To create a Subresource, select the New Resource Group creation.

Sub-resources



Show Filters

Name	Size	Actions
All Resource Groups	1 (0)	
QA	0 (0)	
SIT	1 (1)	
APP	1 (1)	
WEB	1 (1)	

Resource Roles

A role enables you to further refine how a resource is used, and is similar to sub resources. For most deployments, you will not need to define a role.

During process configuration, you select a specific role when determining the resource. A role can be used to set up Serena Release Automation for rolling deployments, balancing, and so on.

For example, you can set up your process as follows:

1. First do a partial deployment to only a small percentage of target machines
2. Next require a manual task for the first role to execute after they have tested the partial deployment
3. Finally, once the manual task has completed, assign the rest of the process to a second role who is responsible for deploying to the rest of the target machines

Role Properties

When you create a role, you can define properties for it. When you add the role to a resource, you can set the values for the properties. For example, if you create a `WS` role and define a `serverURL` property for it, you can access the property like this:

```
${p:resource/WS/serverURL}
```

For information about Serena Release Automation properties, see Chapter 21: Serena Release Automation Properties [page 219].

Agents

An agent is a lightweight process that runs on a target host and communicates with the Serena Release Automation server. Agents perform the actual work of deploying components and so relieves the server from the task, making large deployments involving thousands of targets possible. Usually, an agent runs on the same host where the resources it handles are located; a single agent can handle all the resources on its host. If a host has several resources, an agent process is invoked separately for each one. Depending on the number of hosts in an environment, a deployment might require a large number of agents.

Agents are installed with the batch files provided with the installation files, see Agent Installation [page 63]. Agents that will be installed on Unix machines can also be installed remotely using Serena Release Automation's web application, which is described below. Agents are run using the batch files included with the installation package.

Once an installed agent has been started, the agent opens a socket connection to the Serena Release Automation server (securable by configuring SSL for server-agent communication) based on the information supplied during installation. Agents on networks other than the one where the server is located might need to open a firewall to establish connection. Once communication is established, the agent will be visible in the Serena Release Automation web application where it can be configured. Active agents–regardless of OS–can be upgraded using the web application.

Agent configuration consists of assigning an agent to at least one environment; agents can be assigned to multiple environments. If an agent is assigned to several environments, it can perform work on behalf of all of them.

Remote Agent Installation

You can install an agent onto a Unix machine using the web application. A remotely installed agent cannot be installed as a service.

To install an agent:

1. Display the **Install New Agent** dialog by clicking the **Install New Agent** button on the **Agents** pane `Management > Resources > Agents`.

2. Enter the required information into the dialog's fields:

   Remote Agent Installation fields table

Field	Description
Target Hosts*	Host names or IP addresses of the machines where the agent will be installed.
SSH Port*	SSH port addresses of the machines where the agent will be installed.
SSH Username*	SSH user name used on the machines where the agent will be installed.
Use Public Key Authentication	Check this box if you want to authenticate using public key authentication instead of a password.
SSH Password*	SSH password associated with the user name used on the machines where the agent will be installed.
Agent Name*	Name of the agent.

Field	Description
Agent Dir*	Directory where agent should be installed.
Java Home Path*	Path to Java on the machine where the agent will be installed.
Temp Dir Path*	Path to the directory used to perform the installation on the target machine.
Server Host*	Host name or IP address of the Serena Release Automation server or agent relay to which the agent will connect.
Server Port*	Serena Release Automation server port (7918) or agent relay (7916) to which the agent will connect.
Mutual Authentication	Check this box if the agent should enforce certificate validation for mutual authentication.
Proxy Host	Host name or IP address of the agent relay if used.
Proxy Port	HTTP port of the agent relay (20080) if used.

3. Click **Save** when you are done.

Remotely installed agents will start running automatically. If a remotely installed agent stops running, it must be restarted on the host machine.

Managing Agents Remotely

While we characterize an agent as a process (singular), technically an agent consists of two processes: a *worker* process and a *monitor* process. Worker processes perform the actual work of deployment, such as handling plug-in steps. Monitor processes manage the worker process: handling restarts, upgrades, and tests for example. Once an agent is installed, you can manage (via the monitor process) many of its features from the Serena Release Automation web application. Agent properties can be changed directly by editing the agent's `conf/agent/installed.properties` file and restarting the agent.

To manage an agent:

1. Display the Agents pane `Management > Resources > Agents`.
2. Click an action link for the desired agent. Actions are described in the following table.

   Agent Management table

Action	Description
Edit	This option enables you to edit the agent's description.

Action	Description
Restart	This option will shutdown and restart the agent. While the agent is shutdown, its status will be `Offline`.
Upgrade	This option will shutdown the agent and apply the upgrade. While the agent is shutdown, its status will be `Offline`. After the upgrade is applied, the agent will be restarted. Before its status is `Online`, it might briefly be `Connected`.
Test	This option will perform an agent settings and connection test. Test results are displayed in the **Connection Test** dialog.
Inactivate	This option willl deactivate the agent. Agents that are deactivated cannot perform deployments. To reactivate the agent, check the `Show Inactive Agents` check box on the Agents pane, then click `Activate` for the agent.
Delete	Removes the agent.

Agent Pools

Similar to resource groups, agent pools help you organize and manage agents installed in different environments.

Creating an Agent Pool

To create an agent pool:

1. Display the **Create New Agent Pool** dialog by clicking the **Create New Agent Pool** button on the **Agent Pools** pane `Management > Resources > Agent Pools`.

2. Enter the pool name in the `Name` field.

3. Optional. Enter a description in the `Description` field.

4. Click the `Pool Members` field to add agents to the pool. A selection-type pop-up is displayed listing the available agents.

5. Select the agent or agents you want to add to the pool. Optionally, you can filter the listed agents by entering search text into the text field.

6. When you are finished, click **Save**.

Managing Agent Pools

To manage agent pools:

1. Display the **Agent Pools** pane `Management > Resources > Agent Pools`.

2. Click an action link for the desired pool. Actions are described in the following table.

   Agent Pool Management table

Action	Description
Edit	This option enables you to add/remove agents and edit the pool's name and description.
Copy	Copies (creates a new pool with the same agents as the selected pool) the pool.
Inactivate	This option will deactivate the agent pool.
Delete	Removes the agent pool.

Chapter 17: Configuration and Processes Options

The Serena Release Automation Configuration option enables you to configure applications, their components, and their environments from a single location.

The Processes option enables you to create standalone processes that are independent of components and applications.

- Application/Component/Environment Configuration [page 179]
- Standalone Processes [page 179]

Application/Component/Environment Configuration

You attach properties to elements in the context of an application's hierarchy by using the Configuration page `Management > Configuration > [select element] Add Property [button]`.

Standalone Processes

You can define processes that are not associated with a particular component. You can export them and import them into other standalone processes on the same or other Serena Release Automation systems.

To create a standalone process, navigate to the Create Process dialog `Management > Processes > Create Process [button]`. Enter and save the basic information such as the name and default working directory. Select the process and design it following the procedures in Process Editor [page 135].

From the Processes list you can run, edit, inactivate, and export processes.

- The Export option creates a JSON export file
- You can then import the JSON file into a standalone process on another server

Chapter 18: Deployments

Deployments are typically done with applications (see Creating Applications [page 156] for information about creating applications). To performing a deployment, you run a deployment-type process defined for an application in one of its environments.

An application process is run by the Request Process command on the application's Environment pane `Applications > `*`selected_application`*` > Environment`.

To run an application process:

1. Navigate to the application's Environments pane `Applications >` *`selected_application`* `> Environment`.

2. For the environment where you want to perform the deployment, under **Actions** click **Request Process**.

   The Run Process dialog displays.

3. Select the process you want to run.

4. If you want to use a snapshot, select it from the **Snapshot** drop-down list-box. If you select a snapshot, the deployment will automatically use the component version(s) defined for the snapshot. For information about snapshots, see Snapshots [page 168].

5. If you did not select a snapshot, select a component version from the Version list-box. If more than one component is mapped to the application, each one is listed separately. Version options are described in the following table:
   Table 1. Version Options

Version Option	Description
None	No version for this component. Useful when performing multi-component deployments or testing.
Specific Version	Enables you select any version already in Codestation.
Latest Version	Automatically uses the most recently imported version.
Latest With Status	Automatically uses the most recently imported version with the specified status. Status values are: Latest (default value), Passed Test, Archived.
All With Status	All component versions with the selected status will be deployed. Status values are: Latest, Passed Test, Archived.

Version Option	Description
All in Environment	All component versions already deployed in the environment with the selected status will be deployed. Status values are: Active (default), Staged.
All in Environment (Reversed)	All component versions already deployed in the environment with the selected status will be deployed in reverse order. Status values are: Active (default), Staged.

6. Use the Only Changed Version check box to ensure that only changed versions are deployed (it is checked by default). If checked, no previously deployed versions will be deployed. If, for example, you check the box and select a specific version that was already deployed, the version will *not* be redeployed. Uncheck the box if you want to deploy a version regardless of whether or not it was already deployed (if the inventory is out of date, for instance).

7. Select the process you want to run from the Process list box. All processes created for the application are listed.

8. If you want to run the process at a later time, click the Schedule Deployment? check box (it is unchecked by default). If checked, fields appear enabling you to specify the date and time when the process will run. You can also make the process run on a recurring basis.

9. When finished, click Submit to start the process. An application process will start immediately unless scheduled for a later time.

When a process starts, use the Application Process Request pane to review the deployment's status. This pane is also used if the process requires approvals.

After a process finishes, click the **Details** action to display the Deployment of Component pane, which can be used to review the deployment details.

The actions available for this pane enable you to review the deployment's output log, error log, and input/output parameters.

Deploying Components

Components are typically deployed by application processes. The following table summarizes the steps performed to run an application process. For more information, see Chapter 18: Deployments [page 181].

Deployment Steps table

Step	Description
1. Select environment	Application processes are run at the environment level; you run a process for a particular environment. Selecting an environment automatically selects its agent(s). All processes defined for the application are available.

Step	Description
2. Run processs	You run a process by selecting a process for a given environment and specifying certain other parameters. Processes can also be Chapter 22: Command Line Client (CLI) [page 223], or Scheduling Deployments [page 183].
3. Check results	When a process is started, the Application Process Request pane displays information about the application's status and provides links to logs and the application manifest. If an approval or manual task was used, this pane enables affected users to respond.

Related topics:

- Creating Notification Templates [page 112]
- Configuring Mutual Authentication Mode [page 82]
- Application Gates [page 170]
- Chapter 10: Security Administration [page 93]
- Chapter 19: Reports [page 185]
- Chapter 22: Command Line Client (CLI) [page 223]
- Plug-ins [page 23]
- Adding Agents to a License [page 110]

Scheduling Deployments

Serena Release Automation has a built-in deployment scheduling system for setting regular deployments, or even black-out dates, for your Deployments. Deployments for an individual Application are scheduled on a per-environment basis, set when you run a deployment of a Snapshot or Deployment Process. Black-out dates are set within the individual Environments.

Creating a Schedule

To set up a Scheduled Deployment, go to `Application > Environment > Request Process`. If you are scheduling a Snapshot deployment, go to `Application > Snapshots > Request Process` instead. Regardless of the type of deployment you are scheduling, configuration is the same.

After you check the Schedule Deployment box, Serena Release Automation will prompt you to give the date and time you want the deployment to run. The Make Recurring setting will deploy the Application on a regular schedule. For example, if you are practicing Continuous Delivery, the Daily option will deploy the Application to the target Environment every day.

Once you have scheduled the deployment, it will be added to the Calendar. There, if you click on the Scheduled Deployment, you can edit, delete, or investigate the deployment.

Setting Blackouts

A *blackout* is a set per-environment, per-application. Once set, no deployments (nor snapshots) can be scheduled to occur in that environment. Any previously scheduled deployments to the Environment will fail if they fall within the blackout date you set. To set up a blackout, go to `Application > Environments > Calendar > Add Blackout`). If you need to set blackouts for more than one environment, you must do this for each individual one. Serena Release Automation will prompt you to give the dates and times for the blackout.

Chapter 19: Reports

Serena Release Automation provides deployment- and security-type reports:

Deployment reports

Contain historical information about deployments. Data can be filtered in a variety of ways and reports can be printed and saved. In addition, you can save search criteria for later use (see Deployment Reports [page 186]).

Security reports

Provide information about user roles and privileges (see Security Reports [page 199]).

For information about saving and printing reports (see Saving and Printing Reports [page 202]).

The following tables summarize the out-of-the-box reports.

Deployment Reports table

Report	Description
Deployment Detail	Provides information about deployments executed during a user-specified reporting period. Each report row represents a deployment that executed during the reporting period and matched the filter conditions (see Deployment Detail Report [page 186]).
Deployment Average Duration	Average deployment times for applications executed during a user-specified reporting period (see Deployment Average Duration Report [page 193]).
Deployment Total Duration	Total deployment times for applications executed during a user-specified reporting period (see Deployment Total Duration Report [page 196]).
Deployment Count	Provides information about the number of deployments executed during a user-specified reporting period (see Deployment Count Report [page 189]).

Security Reports table

Report	Description
Application Security	Provides information about user roles and privileges defined for Serena Release Automation-managed applications (see Application Security Report [page 199]).
Component Security	Information about user roles and privileges defined for components (see Component Security Report [page 200]).
Environment Security	Information about user roles and privileges defined for environments (see Environment Security Report [page 201]).
Resource Security	Information about user roles and privileges defined for resources (see Resource Security Report [page 201]).

Deployment Reports

Deployment Reports contain historical information about deployments, such as the total number executed and their average duration. Data can be filtered in a variety of ways and reports can be printed and saved. In addition, you can save search criteria for later use. See Saving and Printing Reports [page 202].

Deployment Detail Report

The *Deployment Detail Report* provides information about deployments executed during a user-specified reporting period. Each report row represents a deployment that executed during the reporting period and matched the filter conditions.

Reports can be filtered in a variety of ways (discussed below), and columns selectively hidden. Reports can be saved and printed. See Saving and Printing Reports [page 202].

When selected, the report runs automatically for the default reporting period–current month–and with all filters set to `Any`. The default report represents all deployments that ran during the current month.

Deployment Detail Fields

Initially, all fields are displayed.

Deployment Detail Fields table

Field	Description
Application	Name of the application that executed the deployment.
Environment	Target environment of the deployment.
Date	Date and time when the deployment was executed.

Field	Description
User	Name of the user who performed the deployment.
Status	Final disposition of the deployment. Possible values are: • `Success` • `Failure` • `Running` • `Scheduled` • `Approval Rejected` • `Awaiting Approval`
Duration	Amount of time the deployment ran. For a successful deployment, the value represents the amount of time taken to complete successfully. If deployment failed to start, no value is given. If a deployment started but failed to complete, the value represents the amount of time it ran before it failed or was cancelled.
Action	This field provides links to additional information about the deployment. The `View Request` link displays the **Application Process Request** pane (see Chapter 15: Managing Applications [page 155].

Running the Deployment Detail Report

To run a report:

1. Use the **Date Range** date-picker to set the report's start- and end-dates.

   Date Range table

Field	Description
Current, Prior Week	Start day is either Sunday or Monday, depending on what is defined in your system.
Current, Prior Month	Start day is first day of the month.
Current, Prior Quarter	Quarters are bound by calendar year.
Current, Prior Year	Current year includes today's date.

Field	Description
Custom	Displays the **Custom** pop-up which enables you to pick an arbitrary start- and end-date.

The start-time is automatically set to 00:00 (24-hour clock) for the selected date in the reporting period.

The end-time is automatically set to 24:00 for the selected date.

2. Set the report filters.

   Filters are set with the properties drop-down list boxes. To set a filter, select it from the corresponding property list box.

   Report Filters table

Field	Description
Application	Only deployments executed by the selected application appear in the report. Default value: `Any`.
Environment	Only deployments executed by the application selected with the **Application** list box that also used this environment appear in the report. If the application value is `Any`, the available value is `Any`; otherwise, environments defined for the selected application are listed.
User	Only deployments executed by the selected user appear in the report. Default value: `Any`.
Status	Only deployments with the selected status appear in the report. Default value: `Any`.
Plugin	Only deployments that used the selected plug-in appear in the report. Default value: `Any`. Note: the `Any` value also includes deployments that did not use a plug-in.

3. Run the report.

   Click the **Run** button to apply your filter conditions to the data and produce the report.

By default, the report is sorted by Application. You can sort the report on any field by clicking on the column header.

For information about saving and printing reports, see Saving and Printing Reports [page 202].

Report Samples: Deployment Detail

The following table contains examples of reports that can be produced using the Deployment Detail Report.

Sample Reports table

Field	Description
Show me: *All failed deployments that occurred on July 4th during the previous year.*	• **Application**: Any • **Status**: Failure • **Date Range**: Use the Custom pop-up to set the start- and end-dates to July 4th.
Show me: *Deployments for an application that used a specific environment.*	• **Application**: Select the value from the drop-down list box. • **Environment**: Select the environment from the drop-down list box. When an application is selected, only environments defined for it are available in the Environment drop-down list box.
Show me: *Failed deployments that used a specific plug-in yesterday.*	• **Status**: Failure • **Plugin**: Select the value from the drop-down list box. • **Date Range**: Use the Custom pop-up to set the start- and end-dates to the previous day.
Show me: *My deployments that used a specific application during the past month.*	• **Application**: Select the value from the drop-down list box. • **User**: Select your user ID. • **Date Range**: Select `Prior Month`.

Deployment Count Report

The *Deployment Count Report* provides information about the number of deployments executed during a user-specified reporting period. The report provides both a tabular presentation and line graph of the data. Each table row represents an environment used by an applications for the reporting period and interval.

The line graph elements are:

- y-axis represents the number of deployments
- x-axis represents reporting intervals

- plot lines represent environments used by applications

The units along the y-axis are scaled to the number of records reported. The units along the x-axis represent the reporting interval, which can be: months, weeks, or days. Each color-coded plot line represents a single environment used by the deployment during the reporting period.

When selected, the report runs automatically for the default reporting period (current month)and reporting interval (days), and with all filters set to `Any`. The default report provides a count of all deployments that ran during the current month.

Deployment Count fields

Deployment Count Fields table

Field	Description
Application	Name of the application that executed the deployment.
Environment	Name of the environment used by the application.
Reporting Interval	The remaining columns display the number of the deployments for the selected reporting interval.

Running the Deployment Count Report

To run a report:

1. Set the reporting period.

   Use the `Date Range` date-picker to set the report's start- and end-dates. The selected value(s) determines the columns in the tabular report, and the coordinate interval on the graph's x-axis. Default value: `Current Month`.

   Date Range

Field	Description
Current, Prior Week	Start day is either Sunday or Monday, depending on what is defined in your system. Reporting interval is set to days.
Current, Prior Month	Start day is first day of the month. Reporting interval is set to days.
Current, Prior Quarter	Quarters are bound by calendar year. Reporting interval is set to weeks.
Current, Prior Year	Reporting interval is set months.

Field	Description
Custom	Displays the Custom pop-up which enables you to pick an arbitrary start- and end-date.

The start-time is automatically set to 00:00 (24-hour clock) for the selected date in the reporting period.

The end-time is automatically set to 24:00 for the selected date.

2. Set the report filters.

   Filters are set with the properties drop-down list boxes. To set a filter, select it from the corresponding property list box.

   Filters table

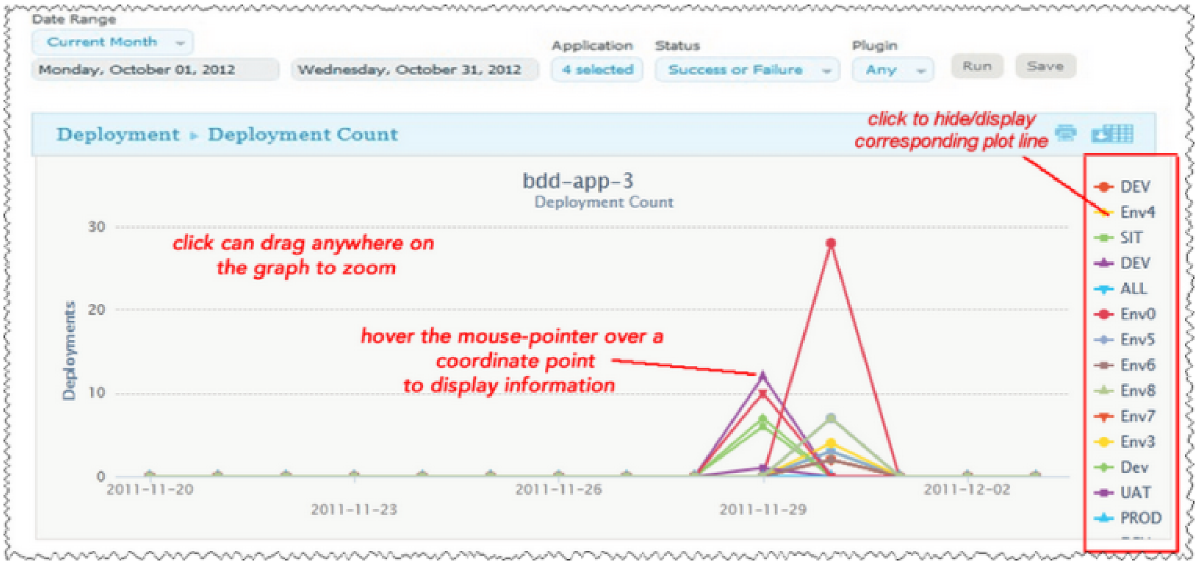
Field	Description
Application	Only deployments executed by the selected application(s) appear in the report. To select applications: a. Click **Application**. b. To include an application in the report, click the corresponding check box. If a large number of applications are listed, type the first few letters of the application's name in the text box to scroll the list. Multiple applications can be selected. c. Click **OK**.
Status	Only deployments with the selected status appear in the report. Default value: `Success or Failure`, which means all deployments.
Plugin	Only deployments that used the selected plug-in appear in the report. Default value: `Any`. Note: the `Any` value also includes deployments that did not use a plug-in.

3. Run the report.

   Click **Run** to apply your filter conditions to the data and produce the report.

A tabular report and line graph are produced.

Deployment Count Graph



Each environment used by a reporting application is represented by an individual plot line and table row. You can hide a plot line by clicking the corresponding item in the graph legend. To see information about a graph coordinate, hover the mouse over the graph point.

You can zoom a graph area by dragging the mouse over the area.

For information about saving and printing reports, see Saving and Printing Reports [page 202].

Report Samples: Deployment Count

The following table contains examples of reports that can be produced using the Deployment Count Report.

Sample Reports

Field	Description
Show me: *The number of successful deployments for two specific applications during the past ten days that used a particular plug-in.*	• **Application**: Select both applications from the Applications dialog. • **Status**: `Success` • **Plugin**: Select the plug-in from the drop-down list box. • **Date Range**: Use the **Custom** pop-up to set the ten-day range.

Field	Description
Show me: *The number of failed deployments for a given application during the past month*	• **Application**: Select the value from the Applications dialog. • **Status**: `Failure` • **Date Range**: Select `Prior Month`.
Show me: *The number of failed deployments that used a specific plug-in yesterday.*	• **Application**: Select the applications from the Applications dialog. • **Status**: `Failure` • **Plugin**: Select the value from the drop-down list box. • **Date Range**: Use the **Custom** pop-up to select the previous day.

Deployment Average Duration Report

The *Deployment Average Duration Report* provides average deployment times for applications executed during a user-specified reporting period. The report provides both a tabular presentation and line graph of the data. Each table row represents an environment used by an application for the reporting period and interval.

The line graph elements are:

- y-axis represents deployment duration average times
- x-axis represents reporting intervals
- plot lines represent environments used by the applications

The units along the y-axis are scaled to the number of records reported. The units along the x-axis represent the reporting interval, which can be: months, weeks, or days. Each color-coded plot line represents a single environment used by the deployment during the reporting period.

When selected, the report runs automatically for the default reporting period (current month)and reporting interval (days), and with all filters set to `Any`. The default report provides average deployment times for all deployments that ran during the current month.

Deployment Average Duration Fields

Average Duration Fields

Field	Description
Application	Name of the application that executed the deployment.
Environment	Name of the environment used by the application.
Reporting Interval	The remaining columns display the average deployment times for the reporting interval.

Running the Deployment Average Duration Report

To run a report:

1. Set the reporting period.

   Use the `Date Range` date-picker to set the report's start- and end-dates. The selected value(s) determines the columns in the tabular report, and the coordinate interval on the graph's x-axis. Default value: `Current Month`.

   Date Range

Field	Description
Current, Prior Week	Start day is either Sunday or Monday, depending on what is defined in your system. Reporting interval is set to days.
Current, Prior Month	Start day is first day of the month. Reporting interval is set to days.
Current, Prior Quarter	Quarters are bound by calendar year. Reporting interval is set to weeks.
Current, Prior Year	Reporting interval is set months.
Custom	Displays the **Custom** pop-up which enables you to pick an arbitrary start- and end-date.

   The start-time is automatically set to 00:00 (24-hour clock) for the selected date in the reporting period.

   The end-time is automatically set to 24:00 for the selected date.

2. Set the report filters.

   Filters are set with the properties drop-down list boxes. To set a filter, select it from the corresponding property list box.

Filters table

Field	Description
Application	Only deployments executed by the selected application(s)appear in the report. To select applications: a. Click the **Application** button. b. To include an application in the report, click the corresponding check box. If a large number of applications are listed, type the first few letters of the application's name in the text box to scroll the list. Multiple applications can be selected. c. Click **OK**.
Status	Only deployments with the selected status appear in the report. Default value: `Success or Failure`, which means all deployments.
Plugin	Only deployments that used the selected plug-in appear in the report. Default value: `Any`. **Note:** `Any` value also includes deployments that did not use a plug-in.

3. Run the report.

   Click **Run** to apply your filter conditions to the data and produce the report.

A tabular report and line graph are produced. Each environment used by a reporting application is represented by an individual plot line and table row. You can hide a plot line by clicking the corresponding item in the graph legend. To see information about a graph coordinate, hover the mouse over the graph point.

You can zoom a graph area by dragging the mouse over the area.

For information about saving and printing reports, see Saving and Printing Reports [page 202].

Sample Reports: Deployment Average Duration

The following table contains examples of reports that can be produced using the Deployment Average Duration Report.

Sample Reports table

Field	Description
Show me: *Average durations for two specific applications during the past ten days that used a particular plug-in.*	• **Application**: Select both applications from the Applications dialog. • **Status**: `Success or Failure` • **Plugin**: Select the plug-in from the drop-down list box. • **Date Range**: Use the Custom pop-up to set the ten-day range.
Show me: *Average durations for successful deployments for a given application during the past six months.*	• **Application**: Select the application from the Applications dialog. • **Status**: `Success` • **Date Range**: Use the Custom pop-up to set the range to the previous six months.

Deployment Total Duration Report

The *Deployment Total Duration Report* provides total deployment times for applications executed during a user-specified reporting period. The report provides both a tabular presentation and line graph of the data. Each table row represents an environment used by one of the selected applications for the reporting period and interval.

The line graph elements are:

- y-axis represents deployment duration times
- x-axis represents reporting intervals
- plot lines represent environments used by the applications

The units along the y-axis are scaled to the number of records reported. The units along the x-axis represent the reporting interval, which can be: months, weeks, or days. Each color-coded plot line represents a single environment used by an application during the reporting period.

When selected, the report runs automatically for the default reporting period (current month)and reporting interval (days), and with all filters set to `Any`. The default report provides total deployment times for all deployments that ran during the current month.

Deployment Total Duration Fields

Total Duration Fields table

Field	Description
Application	Name of the application that executed the deployment.
Environment	Name of the environment used by the application.
Reporting Interval	The remaining columns display the total deployment times for the reporting interval.

Running the Deployment Total Duration Report

To run a report:

1. Set the reporting period.

   Use the `Date Range` date-picker to set the report's start- and end-dates. The selected value(s) determines the columns in the tabular report, and the coordinate interval on the graph's x-axis. Default value: `Current Month`.

   Date Range table

Field	Description
Current, Prior Week	Start day is either Sunday or Monday, depending on what is defined in your system. Reporting interval is set to days.
Current, Prior Month	Start day is first day of the month. Reporting interval is set to days.
Current, Prior Quarter	Quarters are bound by calendar year. Reporting interval is set to weeks.
Current, Prior Year	Reporting interval is set months.
Custom	Displays the **Custom** pop-up which enables you to pick an arbitrary start- and end-date.

   The start-time is automatically set to 00:00 (24-hour clock) for the selected date in the reporting period.

   The end-time is automatically set to 24:00 for the selected date.

2. Set the report filters.

   Filters are set with the properties drop-down list boxes. To set a filter, select it from the corresponding property list box.

Filters table

Field	Description
Application	Only deployments executed by the selected application(s) appear in the report. To select applications: a. Click the **Application** button. b. To include an application in the report, click the corresponding check box. If a large number of applications are listed, type the first few letters of the application's name in the text box to scroll the list. Multiple applications can be selected. c. Click **OK**.
Status	Only deployments with the selected status appear in the report. Default value: `Success or Failure`, which means all deployments.
Plugin	Only deployments that used the selected plug-in appear in the report. Default value: `Any`. **Note:** The `Any` value also includes deployments that did not use a plug-in.

3. Run the report.

   Click the **Run** button to apply your filter conditions to the data and produce the report.

A tabular report and line graph are produced. Each environment used by a reporting application is represented by an individual plot line and table row. You can hide a plot line by clicking the corresponding item in the graph legend. To see information about a graph coordinate, hover the mouse over the graph point.

You can zoom a graph area by dragging the mouse over the area.

For information about saving and printing reports, see Saving and Printing Reports [page 202].

Sample Reports: Deployment Total Duration

The following table contains examples of reports that can be produced using the Deployment Total Duration Report.

Sample Reports table

Field	Description
Show me: *Total duration times for two specific applications during the past ten days that used a particular plug-in.*	• **Application**: Select both applications from the Applications dialog. • **Status**: `Success or Failure` • **Plugin**: Select the plug-in from the drop-down list box. • **Date Range**: Use the Custom pop-up to set the ten-day range.
Show me: *Total duration times for successful deployments for a given application during the past six months.*	• **Application**: Select the application from the **Applications** dialog. • **Status**: `Success` • **Time Unit**: `Months` • **Date Range**: Use the Custom pop-up to set the six-month range.

Security Reports

Security Reports provide information about user roles and privileges defined with the Serena Release Automation security system.

Application Security Report

The Application Security Report provides information about user roles and privileges defined for Serena Release Automation-managed applications. Each report row represents an individual application. When selected, the report runs automatically for all applications.

Application Security Fields

Application Security Fields table

Field	Description
Application	Name of the application.

Field	Description
Run Component Processes	Users who have component process execution privileges. For information about component processes, see Creating Components [page 122].
Execute	Users who have application execution privileges. For information about applications, see Chapter 15: Managing Applications [page 155].
Security	Users who can define privileges for other users. For information about security, see Chapter 10: Security Administration [page 93].
Read	Users who can review information about the application but not change it.
Write	Users who can access and edit the application.

The report is sorted by **Application**. You can change the sort order by clicking on the column header.

For information about saving and printing reports, see Saving and Printing Reports [page 202].

Component Security Report

The Component Security Report provides information about user roles and privileges defined for components. Each report row represents an individual component. When selected, the report runs automatically for all components.

Component Security Fields

Component Security Fields table

Field	Description
Component	Name of the component.
Execute	Users who have component process execution privileges. For information about component processes, see Creating Components [page 122].
Security	Users who can define privileges for other users. For information about security, see Chapter 10: Security Administration [page 93]
Read	Users who can review information about the component but not change it.
Write	Users who can access and edit the component.

The report is sorted by **Component**. You can change the sort order by clicking the column header.

For information about saving and printing reports, see Saving and Printing Reports [page 202].

Environment Security Report

The Environment Security Report provides information about user roles and privileges defined for environments. Each report row represents an individual environment. When selected, the report runs automatically for all environments.

Environment Security Fields

Environment Security Fields table

Field	Description
Application	Name of the application.
Environment	Name of the environment.
Execute	Users who have execution privileges for the environment. For information about environments, see Chapter 15: Managing Applications [page 155].
Security	Users who can define privileges for other users. For information about security, see Chapter 10: Security Administration [page 93].
Read	Users who can review information about the environment (but not change it).
Write	Users who can access and edit the environment.

The report can be sorted by **Application** or **Environment**. By default, it is sorted by **Application**. You can change the sort order by clicking the column header.

For information about saving and printing reports, see Saving and Printing Reports [page 202].

Resource Security Report

The *Resource Security Report* provides information about user roles and privileges defined for resources. Each report row represents an individual resource. When selected, the report runs automatically for all resources.

Resource Security Fields

Resource Security Fields table

Field	Description
Resource	Name of the resource.

Field	Description
Execute	Users who have execution privileges for the resource. For information about resources, see Chapter 16: Managing Resources [page 173].
Security	Users who can define privileges for other users. For information about security, see Chapter 10: Security Administration [page 93].
Read	Users who can review information about the resource but not change it.
Write	Users who can access and edit the resource.

The report is sorted by **Resource**. You can change the sort order by clicking on the column header.

For information about saving and printing reports, see Saving and Printing Reports [page 202].

Saving and Printing Reports

You can print and save report data for all report types. In addition, you can save filter and sort order information for deployment-type reports.

Saving Report Data

Serena Release Automation saves report data in CSV files (comma separated value).

To save report data:

1. Set the filters (if any) and run the report.
2. Click **CSV**.
3. Use the **Opening File** dialog. You can save the data to file, or open the data with an application associated with CSV-type files on your system.

Note: Sort-order and hidden/visible column information is not preserved in the CSV file.

Saving Report Filters

You can save filter and sort-order settings for deployment reports. Saved reports can be retrieved from the `Reports pane > My Reports` menu.

To save a report:

1. Set the filter conditions.
2. Define the reporting period.
3. Run the report.
4. Optional. Set the sort order.

   You can change the sort order for any column by clicking the column header.

5. Optional. Change column visibility. Click **Edit** to display the Select Columns dialog. By default, all columns are selected to appear in a report. To hide a column, click the corresponding check box.
6. Click **Save**. The **Save Current Filters** dialog displays.
7. Enter a name for the file, and save your work.

To run your report, click the report name in the **My Reports** menu.

To delete your report, click **Delete**.

Printing Reports

To print a report:

1. Set the filter conditions.
2. Define the reporting period.
3. Run the report.
4. Optional. Set the sort order.

   Your changes are reflected in the printed report.
5. Optional. Change column visibility. By default, all columns are selected to appear in the printed report. Hidden columns will not appear in the output.
6. Click **Print** to print your report.

Chapter 20: Example—helloWorld Deployment

This section gets you started by providing immediate hands-on experience using key product features. The *helloWorld* walk-through demonstrates how to create a simple deployment using default features.

Note: This section assumes you have installed the Serena Release Automation server and at least one agent. For the walk-through, the agent can be installed on the same machine where the server is installed.

Deployments are done by performing the following steps:

- **Define Components**

  Components represent deployable artifacts: files, images, databases, configuration materials, or anything else associated with a software project. Components have versions which ensure that proper component instances are deployed.

- **Define Component Processes**

  Component processes operate on components, usually by deploying them. Each component must have at least one component process defined for it. For *helloWorld* you will create a single component that contains a number of text-type files (artifacts), and define a simple process that moves—deploys—the artifacts.

- **Define Application**

  An application brings together all deployment components by identifying its components and defining a process to move them through a target environment (by running component processes, for instance).

- **Configure Environment**

  An environment is a collection of resources that represent deployment targets–physical machines, virtual machines, databases, J2EE containers, and so on. Each application must have at least one environment defined for it.

- **Identify Agent**

  Agents are distributed processes that communicate with the Serena Release Automation server and perform the actual work of deploying artifacts. Generally, an agent is installed on the host where the resources it manages reside. Agents are associated with applications at the environment level.

helloWorld: Creating Components

Components contain the artifacts, such as files, images, and databases that you manage and deploy. When creating a component, you:

1. Identify source type.

First, you define the artifacts' source type (all artifacts must be of the same type) and identify where they are stored.

2. Import a version.

   After you identify the artifacts, they are imported into the artifact repository, CodeStation. Artifacts can be imported manually or automatically. When artifacts are imported, they are assigned a version ID, which enables multiple versions to be kept and managed. Snapshots, for example, can employ specific versions.

3. Define a process.

   The process defines how the component artifacts are deployed. A process is designed by assembling plug-in steps. A plug-in step is a basic unit of automation. Steps replace most deployment scripts and/or manual processes. Processes are designed using the drag-and-drop process editor.

helloWorld Deployment

The *helloWorld* deployment moves some files on the local file system to another location on the file system, presumably a location used by an application server. *helloWorld* is a very simple deployment but it uses many key product features—features you will use every day.

Plug-ins provide integration with many common deployment tools and application servers. In addition to Start and Finish steps, each process has at least one step, which can be thought of as a distinct piece of user-configurable automation. By combining steps, complex processes can be easily created. Plug-ins are available for Subversion, Maven, Tomcat, WebSphere, and many other tools.

helloWorld: A Note Before You Begin

You can read the walk-through without actually performing the steps, or you can perform them as you read along. If you want to perform the steps as we go, do the following before starting:

1. Create a directory somewhere on your system and name it: `helloWorld`.

2. Within this `helloWorld` directory, create a sub-directory and name it `hello`.

3. Within the subdirectory create and save five text-type files and name them `artifact1.txt`, `artifact2.txt`, and so on.

   These five files represent the artifacts that you will be deploying. You will be creating a component that contains these files.

Note: If you want to perform the exercise steps, ensure that you have an active agent installed.

helloWorld: Component Version

1. Display the Create Component dialog `Management > Components > Create Component`.

The initially displayed fields are the same for every source type. Other fields appearing depend on the source type and are displayed after a value is selected in the `Source Config Type` field.

2. Enter `helloWorld` in the Name field.

   The name is used when assembling the application. If the component will be used by more than one application, the name should be generic rather than project-specific. For components that are project-specific, a name that conveys something meaningful about the project should be used.

3. Enter a description in the Description field.

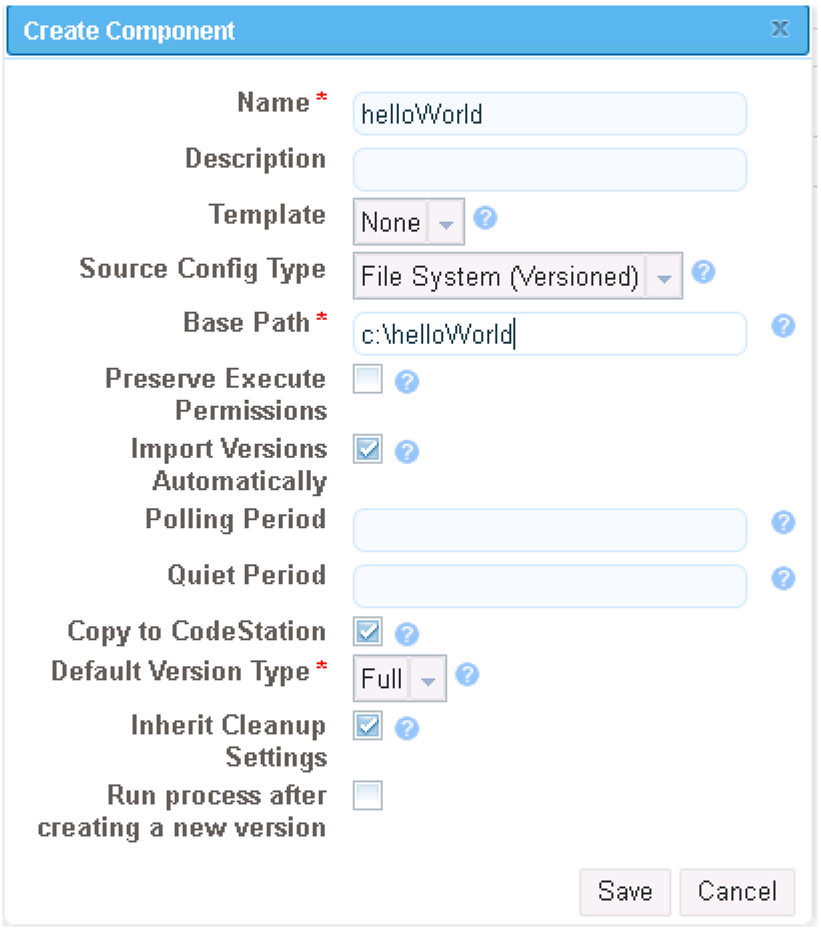
   The optional description can be used to convey additional information about the component. If the component is used by more than one application, for example, entering "Used in applications A and B" can help identify how the component is used. If you are unsure about what to enter, leave the field blank. You can always return to the component and edit the description at any time.

   For this exercise, ignore the Template field. Templates provide a way to reuse commonly used component configurations. For information about templates, see Component Templates [page 148].

4. Select `File System (Versioned)` from the Source Config Type field.

   Selecting a value displays several fields required by the selected type.

   `File System (Versioned)` is one of the simplest configuration options and can be used to quickly set-up a component for evaluation purposes, as we do here.



Create Component

Name *	helloWorld
Description	
Template	None
Source Config Type	File System (Versioned)
Base Path *	c:\helloWorld
Preserve Execute Permissions	☐
Import Versions Automatically	✓
Polling Period	
Quiet Period	
Copy to CodeStation	✓
Default Version Type *	Full
Inherit Cleanup Settings	✓
Run process after creating a new version	☐

5. Complete this option by entering the path to the artifacts.

   In our example, the artifacts are stored inside the subdirectory created earlier. `File System (Versioned)` assumes that subdirectories within the base directory are distinct component versions, which is why we placed the files (artifacts) inside a subdirectory. `File System (Versioned)` can automatically import versions into CodeStation. If a new subdirectory is discovered when the base directory is checked, its content is imported and a new version is created.

6. Check the Import Versions Automatically check box.

   When this option is selected, the source location will be periodically polled for new versions. If this option is not selected, you will have to manually import a new version every time one becomes available.

7. Ensure the Copy to CodeStation check box is selected.

   This option, which is recommended and selected by default, creates a tamper-proof copy of the component's artifacts and stores them in the embedded artifact management system, CodeStation. If you already have an artifact repository, such as AnthillPro or Maven, you might leave the box unchecked.

   For this exercise, you can accept the default values for the remaining options and save your work.

8. After the interval specified by the system settings, the initial component version (the files inside the subdirectory created earlier) should be imported into CodeStation. To ensure the artifacts were imported, display the Versions pane `Management > Components > helloWorld > Versions > version_name`. This pane displays all versions for the selected component. If things went well, the artifacts will be listed. The base path, as you will recall, is `C:\helloWorld`. Within `helloWorld` is the single subdirectory `hello` on my machine).

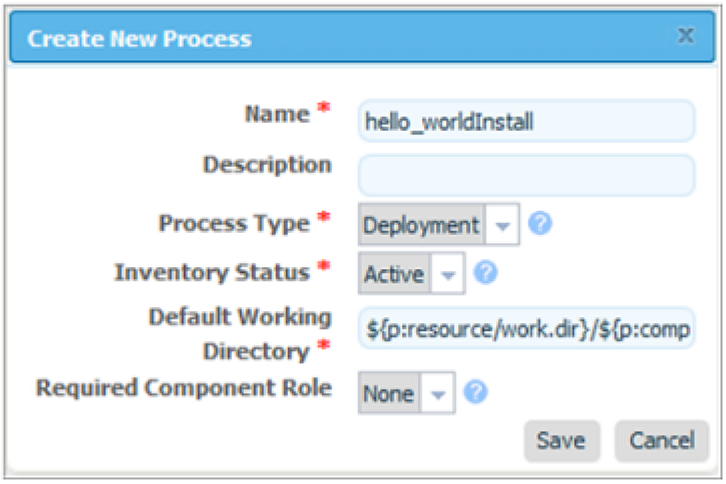
helloWorld: Component Process

Once a component has been created and a version imported, a process to deploy the artifacts—called a component process—is defined.

To Configure the helloWorld Component Process:

1. Display the Create New Process dialog for the `helloWorld` component `Management > Components > helloWorld > Processes > Create New Process`.

   Figure 1. Create New Process dialog


Create New Process

Name *: hello_worldInstall

Description:

Process Type *: Deployment

Inventory Status *: Active

Default Working Directory *: ${p:resource/work.dir}/${p:comp

Required Component Role: None

2. In the Create New Process dialog, enter a name in the Name field.

   The name and description typically reflect the component's content and process type.

3. Enter a meaningful description in the description field.

   If the process will be used by several applications, you can specify that here.

4. Accept the default values for the other fields.

   You might be wondering what the Default Working Directory field does. This field points to the working directory (for temporary files, etc.) used by the agent running the process. The default value resolves to *`agent_directory`*`\work\`*`component_name_directory`*. The default properties work for most components; you might need to change it if a component process cannot be run at the agent's location.

When you finish, save your work.

So far you have created a place-holder for the actual process you will define next. The name you gave the process is listed on the component's Process pane. A component can have any number of processes defined for it.

helloWorld: Process Design

To complete the process you define the actual plug-in steps. In addition to the Start and Finish steps, which are part of every process, a process must have at least one additional step. The steps are defined with the Process Design pane. You define the steps by dropping them onto the design area and arranging them in the order they are to be executed.

To Define the helloWorld Process Steps:

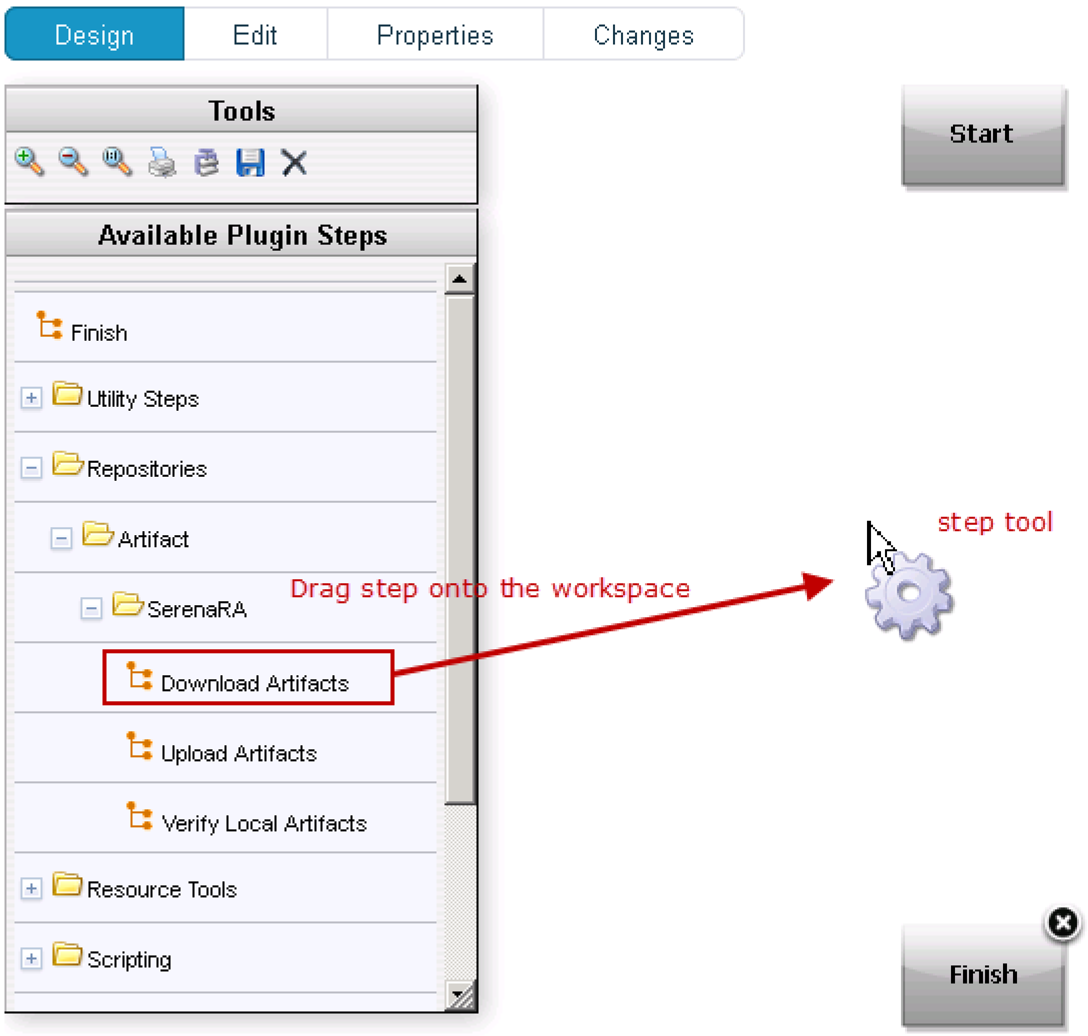
1. Display the Process Design pane for the process created earlier `Management > Components > helloWorld > process_name`.

   The steps are listed in the Available Plug-in Steps list-box. Take a moment to expand the listings and review the available steps. Many commonly used plug-in steps are available out-of-the-box.

2. In the Available Plug-in Steps box, expand the `Serena Release Automation` menu item `Repositories > Artifact > Serena Release Automation`.

3. Drag the `Download Artifacts` step onto the design space and release it. For now, don't worry about where the step is released—a step's position in the workflow is defined after its parameters are configured.

Figure 1. Adding a Step to the Process



The Edit Properties dialog is displayed when the mouse-pointer is released over the design space.

This dialog displays all configurable parameters associated with the selected step.

For this exercise, we can achieve our goal by entering data into a single field—Directory Offset. Recall that the goal for this ambitious deployment is to move the source files in the base directory to another location. As you might guess, several methods for accomplishing this are available. Pointing the Directory Offset field to the target location is one of the simplest.

4. In the Directory Offset field, enter the path to the target directory. Because Serena Release Automation can create a directory during processing, you specify any target directory. I entered `c:\hello` which did not exist on my system, and let Serena Release Automation create it for me.

   If the field is left blank, the process will use the working directory defined earlier. Entering the path overrides the previous value and will cause the source files to be moved—deployed—to the entered location when the process runs. The default value would move (download) the files to `agent_directory\work\component_name_directory`.

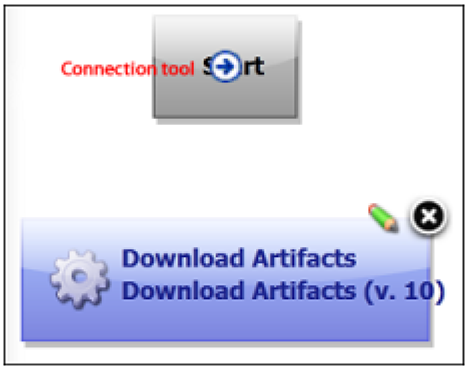
After entering the target path, save your work and close the dialog box.

5. Next, the step must be positioned within the process workflow. There's no requirement that a step be positioned perfectly after it's created; you can place several more before defining their positions, but because this is the only step we are adding, it makes sense to define its position now.

   A step's position in the workflow is defined by dragging connection arrows to/from it. The arrows define the direction of the workflow.

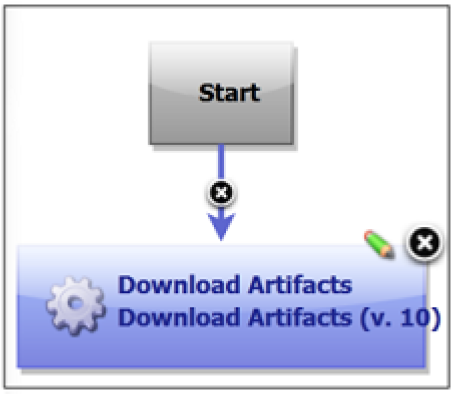
   Hover the mouse pointer over the `Start` step to display the connection tool, as shown in the following illustration. Each step has a connection tool which is used to connect it to other steps.

   Figure 2. Connection Tool

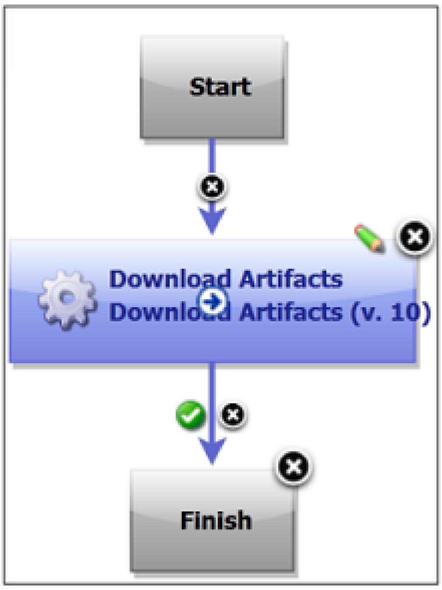
   Grab the connection tool and drag it over the `Download Artifacts` step then release it. A connection arrow connects the two steps. The arrow indicates the direction of process flow—from the originating step to the destination step.

   Figure 3. Finished Connection

6. Complete the process by connecting the `Download Artifacts` step to the `Finish` step. A step can have more than one arrow originating from it and more than one connecting to it.

Figure 4. Completed Process



7. Save the component by using the Save tool on the Tools menu.

Once the process steps are defined, the final task is to define an application that uses the component—and the component process you just created.

helloWorld: Application

To deploy the helloWorld component, you must create an application. An application, as used by Serena Release Automation, is a mechanism that deploys components into environments using *application* processes—processes similar to the component process just defined.

To create an application, you: identify the components it controls (an application can manage any number of components); define at least one environment into which the components will be deployed; and create a process to perform the work. An environment maps components to agents and handles inventory, among other things.

An application process is similar to but not identical to a component process. While application processes consists of steps configured with the process editor, like component processes, they are primarily intended to direct underlying component processes and orchestrate multi-component deployments. The Install Component step, which we will use shortly, enables you to select a component process from among those defined for each component (remember that a component can have more than one process defined for it).

You perform a deployment by running an application process for a specific environment.

You might be wondering why you need to create an application-level process when the process you created for the component should be able to perform the deployment by itself. While individual component processes can be run outside an application process, an environment must still be defined (environments are defined at the application level) and an agent associated with it. For a single-component deployment like *helloWorld*, an application-level process might not be required. You might also want to skip an application-level process when you are testing or patching a component. But for non-trivial deployments, and especially for deployments that have more than one component, you will want to create one or more application-level processes.

helloWorld: Creating an Application

To create an application:

1. Display the Create Application dialog `Management > Applications > Create Application [`*`button`*`]`.

2. Enter a name, for example, `helloWorld_application`, and an (optional) description.

   There are no naming requirements. However the number of associated items, for example components, processes, applications, and environments can become large; so, we recommend you use a naming scheme that makes it easy to identify related items.

3. For the Notification Scheme, accept the default value of `None` from the drop-down, and save the application.

   Serena Release Automation integrates with LDAP and e-mail servers which enables it to send event-based notifications. For example, the default notification scheme will send an e-mail (if an email server has been configured) when a deployment finishes. Notifications can also play a role in deployment approvals (see Creating Notification Templates [page 112]).

Adding the helloWorld Component to the Application

After the application is saved, we identify the component—helloWorld—it will manage. While we have only one component to deploy, an application can manage any number of components.

1. Display the Add a Component dialog for the application just created, helloWorld_application in my case (`Management > Applications > helloWorld_application > Components > Add Component [`*`button`*`]`).

2. Select helloWorld from the Select a Component drop-down list box, then save your selection.

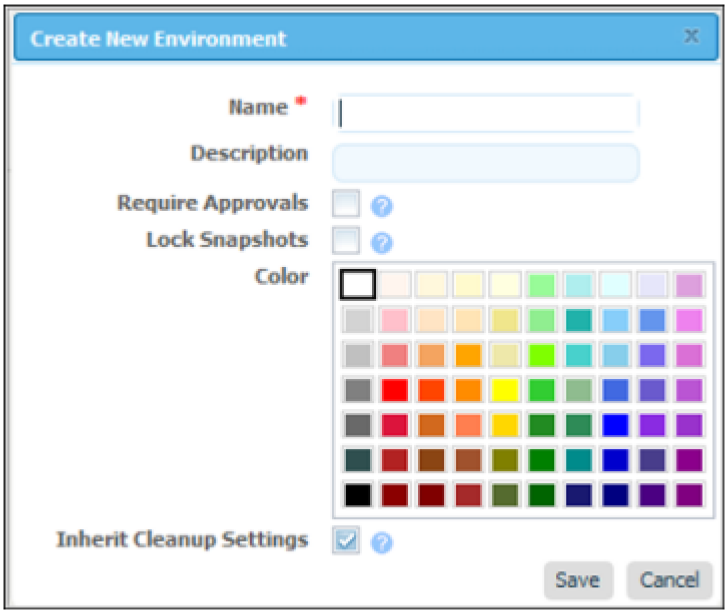
   The simple act of selecting a component accomplishes a lot. The the component processes defined for the component become available to the application, for example, and many application process steps will have default values set to values defined by the helloWorld component.

helloWorld: Adding an Environment to the Application

Before an application can run, it must have at least one environment created for it. An environment defines the agents used by the application, among other things.

1. Display the Create Environment dialog `Management > Applications > helloWorld_application > Create Environment`.

Figure 1. Create Environment



2. Use the Create Environment dialog to define the environment:

   - The value in the Name field will be used in the deployment.

   - If you check the Require Approvals check box, approvals will be enforced. See Chapter 18: Deployments [page 181] for information about the approval process. This is our first deployment so an uncontrolled environment will do fine–leave the box unchecked.

   - Selecting a color provides a visual identifier for the environment in the UI. Typically, each environment will be assigned its own color.

   - Leave the Inherit Cleanup Settings check box checked. Clean-up refers to archiving component versions. When a component is archived, its artifacts are combined into a ZIP file and saved. The corresponding component is removed form CodeStation. When checked, settings are inherited from the system settings, otherwise they are inherited from the application's components.

3. Next, add an agent that will execute the application's process steps. Display the Add a Resource dialog `Applications > [select helloWorld_application] > Environments > [select environment] > Component Mappings > Add a Resource`.

   Select any of the agents that were created when Serena Release Automation was installed.

   While our example uses only a single resource, deployments can use many resources and *resource groups*. Resource groups provide a way to combine resources, which can be useful when multiple deployments use overlapping resources. See Chapter 16: Managing Resources [page 173] for information about resource groups.

helloWorld: Adding a Process to the Application

Now that our application has an environment, we can create an application-level process that will perform the deployment.

1. Display the Create an Application Process dialog `Applications > helloWorld_application > Processes > Create New Process`.
2. Enter a name in the Name field.

   Accept the default values for the other fields:

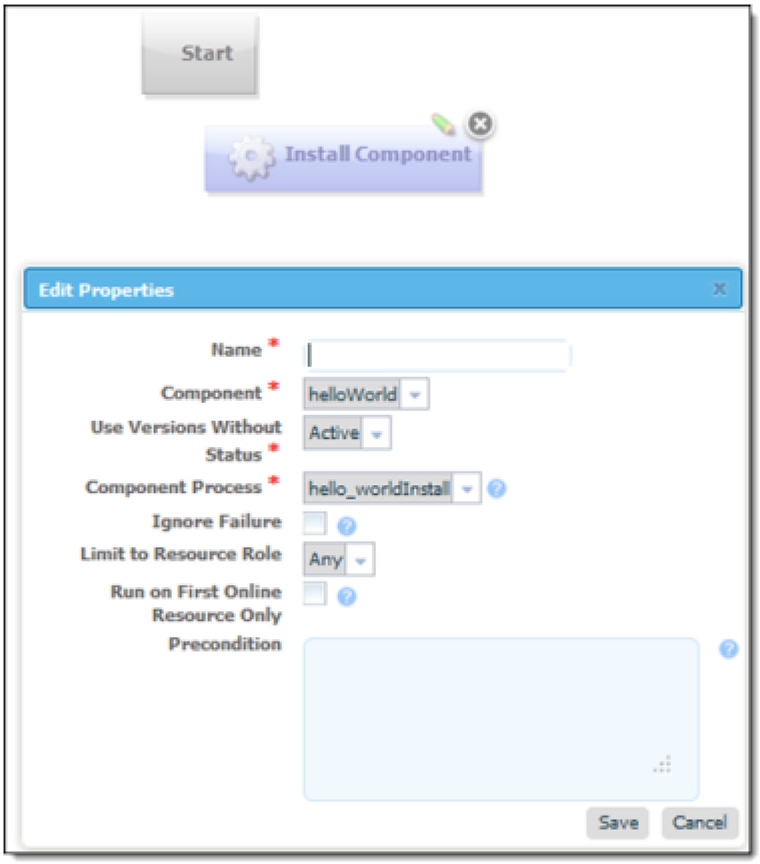
   - The Required Application Role drop-down field is used to restrict who can run this process. The default value, None, means anyone can run the process. The available options are derived from the Serena Release Automation Security System. For information about security roles, see Chapter 10: Security Administration [page 93]
   - The Inventory Management field determines how inventory for the application's components are handled. If you want to manually control inventory, you would select the `Advanced` option. See Statuses [page 91], for information about inventory management.
3. Save your work when you are finished.

Designing the Process Steps

To create an application-level process, you define the individual steps as you did earlier when you used the Process Design pane to create the helloWorld component process (see Component Processes [page 134]).

1. Display the Process Design pane `Applications > application_name > Processes > process_name`. The out-of-box process steps are listed in the Add a Component Process list box.
2. Drag the `Install Component` step onto the design area and release. The Edit Properties dialog is displayed.

Figure 1. Edit Properties dialog



Name *

Component * helloWorld

Use Versions Without Status * Active

Component Process * hello_worldInstall

Ignore Failure

Limit to Resource Role Any

Run on First Online Resource Only

Precondition

Select a component from the **Component** drop-down list box. If you followed the *Quick Start*, the `helloWorld` component will be listed.

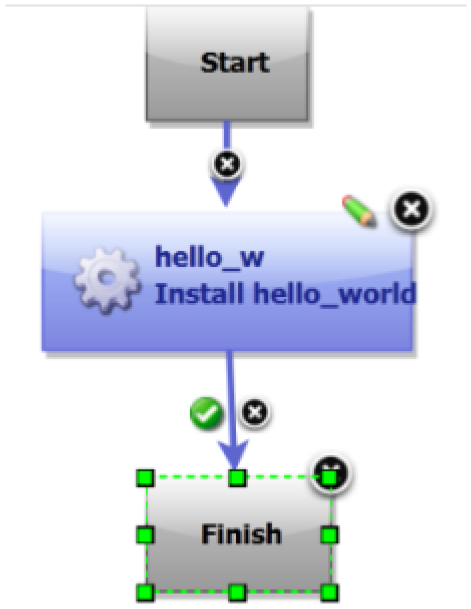
If we wanted this application to install multiple components, we could add a separate Install Component step for each one.

3. Use the Component Process list box to select the component process you created earlier. All processes defined for the selected component are listed. If the component had another process that deployed it to a different location, you could add another Install Component step that used that process—simultaneously installing the component into two different locations.

   Accept the default values for the other fields (see Chapter 15: Managing Applications [page 155] for information about the other fields), and click **Save**.

4. Connect the step to the `Start` and `Finish` steps.

Figure 2. Finished Application Process



5. Save the process by clicking the Save tool on the Tools bar.

Running the Application

Now that the component, environment, and process are complete, you can run the application.

1. On the Application pane, click the Request Process button for the environment you created earlier.
2. On the Run Process dialog:
   a. Select the process you created from the Process drop-down list box. Applications can have more than one process defined for them.
   b. Select Latest Version from the Version drop-down list box. This option ensures that the latest (or first and only) version is affected.
3. Click Submit to run the application.

   The Application Process pane is displayed. This pane displays the application's status.

   Take a few moments to examine the information on this pane; hopefully, you will see a Success message.
4. To see additional information (Output Log, Error Log, Application Properties), click the Details link.

Chapter 21: Serena Release Automation Properties

A step has access to properties:

- set earlier by other steps within the process
- set by the application that invoked the component process
- set for the target environment and resource

Step property values become unavailable once the component process ends.

Every item from the table below will use this format: `${p:version.name}`

Serena Release Automation Properties table

Property	**Description**
version.name	A user defined name to distinguish the version from others. A version name is entered when a new version is imported.
version.id	The number assigned to the version. A version id is created when a new version is imported in CodeStation.
component.name	A user defined name to distinguish it from other components. A component name is entered when creating a new component.
component.id	A unique number Serena Release Automation assigns to distinguish the component from others. The component id is created when a component is created in Serena Release Automation.
resource.name	A user defined name to distinguish it from other resources. The resource name is entered when editing or creating a new resource.
resource.id	A unique number given to a resource. A resource id is assigned when a new resource is created.
application.name	A user defined name to distinguish it from others. An application name is entered when editing or creating a new application.

Property	Description
application.id	A unique number given to an application. An application id is assigned when a new application is created in Serena Release Automation.
environment.name	A user defined name to distinguish the environment from others. An environment name is entered when editing or creating a new environment.
environment.id	A unique number given to an environment. An environment id is assigned when a new environment is created.
agent.id	A unique number Serena Release Automation gives the agent to distinguish it from others with similar names. An agent id is assigned when it is installed on the system.
agent.name	A user defined name to distinguish the agent from others. The agents name can be entered by editing the agent's `conf/agent/installed.properties` file and restarting the agent.
stepname/propertyname	Used in output from step
property_name	Component or application process property; defined on the process's Properties tab. Given value by whoever runs the process.
component/*property_name*	Component custom property; set on the component's Properties tab.
environment/*property_name*	Environment property. Defined on the component's or environment's Properties tab. While both use the same syntax, the latter is not associated with any specific component. Values are supplied on the associated environment or component. A value set on component environment overrides one with the same name set directly on an environment property. Component environment properties are versioned and enable you to centralize properties, tracking type and default values, for instance. Environment properties are unversioned and provide ad-hoc lists of property=value pairs.
resource/*property_name*	Resource properties. This can include the built-in agent properties as well as any custom properties. Each of these have their own tab on the resource.

Property	Description
resource/role name/ property name	Resource role properties. These are defined on resource roles, and the values are set when you add a role to a resource.
application/property name	Application custom properties. These are set on the application's properties tab.
system/property name	Global system properties. These are set on the System Properties page in the Settings area.

All of the following are comma-separated series of name=value, including each property on the given object. This is useful for token replacement.

Name/Value Pairs table

Property	Description
component/ allProperties	Selects all the properties with the same value in a given component.
environment/ allProperties	Selects all the properties with the same value in a given environment.
resource/allProperties	Selects all properties with the same value in a given resource.
system/allProperties	Selects all properties with the same value in a given system.

Chapter 22: Command Line Client (CLI)

CLI is a command-line interface that provides access to the Serena Release Automation server. It can be used to find or set properties, and perform numerous functions, as described in the following topics.

Command Format

To perform a command, open a command window and invoke `serenara-client` along with the command and parameters. Command's have the following format:

```
    serenara-client [global-args...] [global-flags...]
    <command> [args...]
```

The global arguments are:

Table 1. Glogal Arguments

Argument	Description
-authtoken, -authtoken	Optional. Can be set via the environment variable DS_AUTH_TOKEN. An authentication token generated by the server. Either an authtoken or a username and password is required.
-password, -password	Optional. Can be set via the environment variable DS_PASSWORD. A password to authenticate with the server. Either an authtoken or a username and password is required.
-username, -username	Optional. Can be set via the environment variable DS_USERNAME. A username to authenticate with the server. Either an authtoken or a username and password is required.
-weburl, -weburl	Required. Can be set via the environment variable DS_WEB_URL. The base URL of the Serena Release Automation server— http://ds.domain.com:8585.

The global flags are:

Table 2. Global Flags

Flag	Description
`-t, –getTemplate`	Show the JSON template for the command instead of running the command. If a file argument is provided, the template will be output to that file.
`-h, –help`	Print the full description and help of the given command instead of running the command.
`-v, –verbose`	Print extra information during execution.

Note: CLI commands and parameters are case sensitive.

Here is an example using the `getResources` command:

```
serenara-client -weburl http://localhost:8080 -username admin -password admin getResou
```

Commands

Note: CLI commands do not support new lines. Sample entries are broken for display purposes only.

addActionToRoleForApplications

Add action to a role for applications.

Format

```
serenara-client [global-args...] [global-flags...]
addActionToRoleForApplications [args...]
```

Options

```
-role, --role
   Required. Name of the role


-action, --action
   Required. Name of the action
```

addActionToRoleForComponents

Add action to a role for components

Format

```
serenara-client [global-args...] [global-flags...]
addActionToRoleForComponents [args...]
```

Options

```
-role, --role
   Required. Name of the role
```

```
-action, --action
   Required. Name of the action
```

addActionToRoleForEnvironments

Add action to a role for environments

Format

```
serenara-client [global-args...] [global-flags...]
addActionToRoleForEnvironments [args...]
```

Options

```
-role, --role
   Required. Name of the role
```

```
-action, --action
   Required. Name of the action
```

addActionToRoleForResources

Add action to a role for resources

Format

```
serenara-client [global-args...] [global-flags...]
addActionToRoleForResources [args...]
```

Options

```
-role, --role
   Required. Name of the role
```

```
-action, --action
   Required. Name of the action
```

addActionToRoleForUI

Add action to a role for the UI

Format

```
serenara-client [global-args...] [global-flags...]
addActionToRoleForUI [args...]
```

Options

```
-role, --role
   Required. Name of the role


-action, --action
   Required. Name of the action
```

addComponentToApplication

Add a component to an Application.

Format

```
serenara-client [global-args...] [global-flags...]
addComponentToApplication [args...]
```

Options

```
-component, --component
   Required. Name of the component to add


-application, --application
   Required. Name of the application to add it to.
```

addGroupToRoleForApplication

Add a group to a role for an application

Format

```
serenara-client [global-args...] [global-flags...]
addGroupToRoleForApplication [args...]
```

Options

```
-group, --group
   Required. Name of the group


-role, --role
   Required. Name of the role


-application, --application
   Required. Name of the application
```

addGroupToRoleForComponent

Add a group to a role for a component

Format

```
serenara-client [global-args...] [global-flags...]
addGroupToRoleForComponent [args...]
```

Options

```
-group, --group
   Required. Name of the group


-role, --role
   Required. Name of the role


-component, --component
   Required. Name of the component
```

addGroupToRoleForEnvironment

Add a group to a role for an environment

Format

```
serenara-client [global-args...] [global-flags...]
addGroupToRoleForEnvironment [args...]
```

Options

```
-group, --group
   Required. Name of the group


-role, --role
   Required. Name of the role


-application, --application
   Required. Name of the application


-environment, --environment
   Required. Name of the environment
```

addGroupToRoleForResource

Add a group to a role for a resource

Format

```
serenara-client [global-args...] [global-flags...]
addGroupToRoleForResource [args...]
```

Options

```
-group, --group
   Required. Name of the group


-role, --role
   Required. Name of the role


-resource, --resource
   Required. Name of the resource
```

addGroupToRoleForUI

Add a group to a role for the UI.

Format

```
serenara-client [global-args...] [global-flags...]
addGroupToRoleForUI [args...]
```

Options

```
-group, --group
   Required. Name of the group


-role, --role
   Required. Name of the role
```

addLicense

Add a license to the server.

Format

```
serenara-client [global-args...] [global-flags...]
addLicense [args...]
```

Options

No options for this command.

addNameConditionToGroup

Add a name condition to a resource group. Only works with dynamic groups.

Format

```
serenara-client [global-args...] [global-flags...]
addNameConditionToGroup [args...]
```

Options

```
-comparison, --comparison
   Required. Type of the comparison


-value, --value
   Required. Value of the comparison


-group, --group
   Required. Path of the parent resource group
```

addPropertyConditionToGroup

Add a property condition to a resource group. Only works with dynamic groups.

Format

```
 serenara-client [global-args...] [global-flags...]
 addPropertyConditionToGroup [args...]
```

Options

```
-property, --property
   Required. Name of the property


-comparison, --comparison
   Required. Type of the comparison


-value, --value
   Required. Value of the comparison


-group, --group
   Required. Path of the parent resource group
```

addResourceToGroup

Add a resource to a resource group. Only works with static groups.

Format

```
serenara-client [global-args...] [global-flags...]
addResourceToGroup [args...]
```

Options

```
-resource, --resource
   Required. Name of the resource to add
```

```
-group, --group
   Required. Path of the resource group to add to
```

addRoleToResource

Add a role to a resource.

Format

```
serenara-client [global-args...] [global-flags...]
addRoleToResource [args...]
```

Options

```
-resource, --resource
   Required. Name of the parent resource.


-role, --role
   Required. Name of the new resource.
```

addRoleToResourceWithProperties

Add a role to a resource. This command takes a JSON request body. Use the `-t` flag to view the template for the data required for this command.

Format

```
serenara-client [global-args...] [global-flags...]
addRoleToResourceWithProperties [args...] [-] [filename]

-
   Read JSON input from the stdin. See command for requirements.

filename
   Read JSON input from a file with the given filename. See command for requirements.
```

Options

```
No options for this command.
```

addUserToGroup

Add a user to a group

Format

```
serenara-client [global-args...] [global-flags...]
addUserToGroup [args...]
```

Options

```
-user, --user
   Required. Name of the user


-group, --group
   Required. Name of the group
```

addUserToRoleForApplication

Add a user to a role for an application

Format

```
serenara-client [global-args...] [global-flags...]
addUserToRoleForApplication [args...]
```

Options

```
-user, --user
   Required. Name of the user


-role, --role
   Required. Name of the role


-application, --application
   Required. Name of the application
```

addUserToRoleForComponent

Add a user to a role for a component

Format

```
serenara-client [global-args...] [global-flags...]
addUserToRoleForComponent [args...]
```

Options

```
-user, --user
   Required. Name of the user


-role, --role
   Required. Name of the role


-component, --component
   Required. Name of the component
```

addUserToRoleForEnvironment

Add a user to a role for an environment

Format

```
serenara-client [global-args...] [global-flags...]
addUserToRoleForEnvironment [args...]
```

Options

```
-user, --user
   Required. Name of the user

-role, --role
   Required. Name of the role

-application, --application
   Required. Name of the application

-environment, --environment
   Required. Name of the environment
```

addUserToRoleForResource

Add a user to a role for a resource

Format

```
serenara-client [global-args...] [global-flags...]
addUserToRoleForResource [args...]
```

Options

```
-user, --user
   Required. Name of the user

-role, --role
   Required. Name of the role

-resource, --resource
   Required. Name of the resource
```

addUserToRoleForUI

Add a user to a role for the UI

Format

```
serenara-client [global-args...] [global-flags...]
addUserToRoleForUI [args...]
```

Options

```
-user, --user
   Required. Name of the user


-role, --role
   Required. Name of the role
```

addVersionFiles

Upload files to a version

Format

```
serenara-client [global-args...] [global-flags...]
addVersionFiles [args...]
```

Options

```
-component, --component
   Optional. Name/ID of the component (Only required if not using version ID)


-version, --version
   Required. Name/ID of the version


-base, --base
   Required. Local base directory for upload.
   All files inside this will be sent.


-offset, --offset
   Optional. Target path offset (the directory in the version files
   to which these files should be added)
```

addVersionStatus

Add a status to a version

Format

```
serenara-client [global-args...] [global-flags...]
addVersionStatus [args...]
```

Options

```
    -component, --component
       Optional. Name/ID of the component (Only required if not using version ID)


    -version, --version
       Required. Name/ID of the version


    -status, --status
       Required. Name of the status to apply
```

createApplication

Create a new application. This command takes a JSON request body. Use the `-t` flag to view the template for the data required for this command.

Format

```
serenara-client [global-args...] [global-flags...]
createApplication [args...] [-] [filename]


-
   Read JSON input from the stdin. See command for requirements.


filename
   Read JSON input from a file with the given filename. See command for requirements.
```

Options

```
    No options for this command.
```

createApplicationProcess

Create a new application process. This command takes a JSON request body. Use the -t flag to view the template for the data required for this command.

Format

```
serenara-client [global-args...] [global-flags...]
createApplicationProcess [args...] [-] [filename]


-
   Read JSON input from the stdin. See command for requirements.


filename
   Read JSON input from a file with the given filename. See command for requirements.
```

Options

```
    No options for this command.
```

createComponent

Create a new component. This command takes a JSON request body. Use the `-t` flag to view the template for the data required for this command.

Format

```
serenara-client [global-args...] [global-flags...]
createComponent [args...] [-] [filename]


-
   Read JSON input from the stdin. See command for requirements.


filename
   Read JSON input from a file with the given filename. See command for requirements.
```

Options

```
No options for this command.
```

createComponentProcess

Create a new component process. This command takes a JSON request body. Use the `-t` flag to view the template for the data required for this command.

Format

```
serenara-client [global-args...] [global-flags...]
createComponentProcess [args...] [-] [filename]


-
   Read JSON input from the stdin. See command for requirements.


filename
   Read JSON input from a file with the given filename. See command for requirements.
```

Options

```
    No options for this command.
```

createDynamicResourceGroup

Create a new static resource group.

Format

```
serenara-client [global-args...] [global-flags...]
createDynamicResourceGroup [args...]
```

Options

```
-path, --path
   Required. Path to add the resource group to (parent resource group path).

-name, --name
   Required. Name of the new resource group.
```

createEnvironment

Create a new environment.

Format

```
serenara-client [global-args...] [global-flags...]
createEnvironment [args...]
```

Options

```
-application, --application
   Required. Application to add the environment to.

-name, --name
   Required. Name of the new environment.

-description, --description
   Optional. Description of the new environment.

-color, --color
   Optional. Color of the new environment.

-requireApprovals, --requireApprovals
   Optional. Does the environment require approvals?
```

createGroup

Add a new group

Format

```
serenara-client [global-args...] [global-flags...]
createGroup [args...]
```

Options

```
-group, --group
   Required. Name of the group
```

createMapping

Create a new mapping.

Format

```
serenara-client [global-args...] [global-flags...]
createMapping [args...]
```

Options

```
-environment, --environment
   Required. The environment for the mapping.


-component, --component
   Required. The component for the mapping.


-resourceGroupPath, --resourceGroupPath
   Required. The resource group for the apping.


-application, --application
   Optional. The application for the mapping.
   Only necesary if specifying env name instead of id.
```

createResourceGroup

Create a new static resource group.

Format

```
serenara-client [global-args...] [global-flags...]
createResourceGroup [args...]
```

Options

```
-path, --path
   Required. Path to add the resource group to (parent resource group path).


-name, --name
   Required. Name of the new resource group.
```

createRoleForApplications

Create a role for applications

Format

```
serenara-client [global-args...] [global-flags...]
createRoleForApplications [args...]
```

Options

```
-role, --role
   Required. Name of the role
```

createRoleForComponents

Create a role for components

Format

```
serenara-client [global-args...] [global-flags...]
createRoleForComponents [args...]
```

Options

```
-role, --role
   Required. Name of the role
```

createRoleForEnvironments

Create a role for environments

Format

```
serenara-client [global-args...] [global-flags...]
createRoleForEnvironments [args...]
```

Options

```
-role, --role
   Required. Name of the role
```

createRoleForResources

Create a role for resources

Format

```
serenara-client [global-args...] [global-flags...]
createRoleForResources [args...]
```

Options

```
-role, --role
   Required. Name of the role
```

createRoleForUI

Create a role for the UI

Format

```
    serenara-client [global-args...] [global-flags...]
    createRoleForUI [args...]
```

Options

```
    -role, --role
       Required. Name of the role
```

createSubresource

Create a new subresource.

Format

```
    serenara-client [global-args...] [global-flags...]
    createSubresource [args...]
```

Options

```
    -parent, --parent
       Required. Name of the parent resource.


    -name, --name
       Required. Name of the new resource.


    -description, --description
       Optional. Description of the resource.
```

createUser

Add a new user This command takes a JSON request body. Use the -t flag to view the template for the data required for this command.

Format

```
serenara-client [global-args...] [global-flags...]
createUser [args...] [-] [filename]

-
   Read JSON input from the stdin. See command for requirements.

filename
   Read JSON input from a file with the given filename.
   See command for requirements.
```

Options

```
No options for this command.
```

createVersion

Create a new version for a component

Format

```
serenara-client [global-args...] [global-flags...]
createVersion [args...]
```

Options

```
-component, --component
   Required. Name/ID of the component


-name, --name
   Required. Name of the new version
```

deleteGroup

Delete a group

Format

```
serenara-client [global-args...] [global-flags...]
deleteGroup [args...]
```

Options

```
-group, --group
   Required. Name of the group
```

deleteResourceGroup

null

Format

```
serenara-client [global-args...] [global-flags...]
deleteResourceGroup [args...]
```

Options

```
-group, --group
   Required. Path of the resource group to delete
```

deleteResourceProperty

Remove a custom property from a resource

Format

```
serenara-client [global-args...] [global-flags...]
deleteResourceProperty [args...]
```

Options

```
-resource, --resource
   Required. Name of the resource to configure


-name, --name
   Required. Name of the property
```

deleteUser

Delete a user

Format

```
serenara-client [global-args...] [global-flags...]
deleteUser [args...]
```

Options

```
-user, --user
   Required. Name of the user
```

exportGroup

Add a new group

Format

```
serenara-client [global-args...] [global-flags...]
exportGroup [args...]
```

Options

```
-group, --group
   Required. Name of the group
```

getApplication

Get a JSON representation of an application

Format

```
serenara-client [global-args...] [global-flags...]
getApplication [args...]
```

Options

```
-application, --application
    Required. Name of the application to look up.
```

getApplicationProcess

Get a JSON representation of an Application Process

Format

```
serenara-client [global-args...] [global-flags...]
getApplicationProcess [args...]
```

Options

```
-application, --application
    Required. Name of the application

-applicationProcess, --applicationProcess
    Required. Name of the process
```

getApplicationProcessRequestStatus

Get the status for an application request.

Format

```
serenara-client [global-args...] [global-flags...]
getApplicationProcessRequestStatus [args...]
```

Options

```
-request, --request
    Required. ID of the application process request to view
```

getApplications

Get a JSONArray representation of all applications

Format

```
serenara-client [global-args...] [global-flags...]
getApplications [args...]
```

Options

```
No options for this command.
```

getComponent

Get a JSON representation of a component

Format

```
serenara-client [global-args...] [global-flags...]
getComponent [args...]
```

Options

```
-component, --component
    Required. Name of the component to look up
```

getComponentProcess

Get a JSON representation of a componentProcess

Format

```
serenara-client [global-args...] [global-flags...]
getComponentProcess [args...]
```

Options

```
-component, --component
    Required. Name of the component

-componentProcess, --componentProcess
    Required. Name of the component
```

getComponents

Get a JSONArray representation of all components

Format

```
serenara-client [global-args...] [global-flags...]
getComponents [args...]
```

Options

```
No options for this command.
```

getComponentsInApplication

Get all components in an application

Format

```
serenara-client [global-args...] [global-flags...]
getComponentsInApplication [args...]
```

Options

```
-application, --application
    Required. Name of the application to get components for
```

getEnvironment

Get a JSON representation of an environment

Format

```
serenara-client [global-args...] [global-flags...]
getEnvironment [args...]
```

Options

```
-environment, --environment
    Required. Name of the environment to look up
```

getEnvironmentsInApplication

Get all environments in an application

Format

```
serenara-client [global-args...] [global-flags...]
getEnvironmentsInApplication [args...]
```

Options

```
-application, --application
    Required. Name of the application to get environments for
```

getMapping

Get a JSON representation of a mapping

Format

```
serenara-client [global-args...] [global-flags...]
getMapping [args...]
```

Options

```
-mapping, --mapping
   Required. ID of the mapping to look up
```

getResource

Get a JSON representation of a resource

Format

```
serenara-client [global-args...] [global-flags...]
getResource [args...]
```

Options

```
-resource, --resource
   Required. Name of the resource to look up
```

getResourceGroup

Get a JSON representation of a resource group

Format

```
serenara-client [global-args...] [global-flags...]
getResourceGroup [args...]
```

Options

```
-group, --group
   Required. Path of the resource group to show
```

getResourceGroups

Get a JSONArray representation of all resource groups

Format

```
serenara-client [global-args...] [global-flags...]
getResourceGroups [args...]
```

Options

```
No options for this command.
```

getResourcesInGroup

Get a JSONArray representation of all resources in a group

Format

```
serenara-client [global-args...] [global-flags...]
getResourcesInGroup [args...]
```

Options

```
-group, –group
   Required. Path of the resource group
```

getResources

Get a JSONArray representation of all resources

Format

```
serenara-client [global-args...] [global-flags...]
getResources [args...]
```

Options

```
No options for this command.
```

getResourceProperty

Get the value of a custom property on a resource

Format

```
serenara-client [global-args...] [global-flags...]
getResourceProperty [args...]
```

Options

```
-resource, --resource
   Required. Name of the resource

-name, --name
   Required. Name of the property
```

getRoleForApplications

Get a JSON representation of a role

Format

```
serenara-client [global-args...] [global-flags...]
getRoleForApplications [args...]
```

Options

```
-role, --role
   Required. Name of the role
```

getRoleForComponents

Get a JSON representation of a role

Format

```
serenara-client [global-args...] [global-flags...]
getRoleForComponents [args...]
```

Options

```
-role, --role
   Required. Name of the role
```

getRoleForEnvironments

Get a JSON representation of a role

Format

```
serenara-client [global-args...] [global-flags...]
getRoleForEnvironments [args...]
```

Options

```
-role, --role
   Required. Name of the role
```

getRoleForResources

Get a JSON representation of a role

Format

```
serenara-client [global-args...] [global-flags...]
getRoleForResources [args...]
```

Options

```
-role, --role
   Required. Name of the role
```

getRoleForUI

Get a JSON representation of a role

Format

```
serenara-client [global-args...] [global-flags...]
getRoleForUI [args...]
```

Options

```
-role, --role
   Required. Name of the role
```

getUser

Get a JSON representation of a user

Format

```
serenara-client [global-args...] [global-flags...]
getUser [args...]
```

Options

```
-user, --user
   Required. Name of the user
```

importGroup

Add a new group This command takes a JSON request body. Use the -t flag to view the template for the data required for this command.

Format

```
serenara-client [global-args...] [global-flags...]
importGroup [args...] [-] [filename]

-
   Read JSON input from the stdin. See command for requirements.

filename
   Read JSON input from a file with the given filename. See command for requirements.
```

Options

```
No options for this command.
```

importVersions

Run the source config integration for a component This command takes a JSON request body. Use the -t flag to view the template for the data required for this command.

Format

```
serenara-client [global-args...] [global-flags...]
importVersions [args...] [-] [filename]

-
   Read JSON input from the stdin. See command for requirements.

filename
   Read JSON input from a file with the given filename. See command for requirements.
```

Options

```
No options for this command.
```

login

Login for further requests

Format

```
serenara-client [global-args...] [global-flags...]
login [args...]
```

Options

```
 No options for this command.
```

logout

Logout

Format

```
serenara-client [global-args...] [global-flags...]
logout [args...]
```

Options

```
No options for this command.
```

removeActionFromRoleForApplications

Add action to a role for applications

Format

```
serenara-client [global-args...] [global-flags...]
removeActionFromRoleForApplications [args...]
```

Options

```
-role, --role
   Required. Name of the role

-action, --action
   Required. Name of the action
```

removeActionFromRoleForComponents

Add action to a role for components

Format

```
serenara-client [global-args...] [global-flags...]
removeActionFromRoleForComponents [args...]
```

Options

```
-role, --role
   Required. Name of the role

-action, --action
   Required. Name of the action
```

removeActionFromRoleForEnvironments

Add action to a role for environments

Format

```
serenara-client [global-args...] [global-flags...]
removeActionFromRoleForEnvironments [args...]
```

Options

```
-role, --role
   Required. Name of the role

-action, --action
   Required. Name of the action
```

removeActionFromRoleForResources

Add action to a role for resources

Format

```
serenara-client [global-args...] [global-flags...]
removeActionFromRoleForResources [args...]
```

Options

```
-role, --role
   Required. Name of the role

-action, --action
   Required. Name of the action
```

removeActionFromRoleForUI

Add action to a role for the UI

Format

```
serenara-client [global-args...] [global-flags...]
removeActionFromRoleForUI [args...]
```

Options

```
-role, --role
   Required. Name of the role

-action, --action
   Required. Name of the action
```

removeGroupFromRoleForApplication

Remove a group to a role for an application

Format

```
serenara-client [global-args...] [global-flags...]
removeGroupFromRoleForApplication [args...]
```

Options

```
-group, --group
   Required. Name of the group

-role, --role
   Required. Name of the role

-application, --application
   Required. Name of the application
```

removeGroupFromRoleForComponent

Remove a group to a role for a component

Format

```
serenara-client [global-args...] [global-flags...]
removeGroupFromRoleForComponent [args...]
```

Options

```
-group, --group
   Required. Name of the group

-role, --role
   Required. Name of the role

-component, --component
   Required. Name of the component
```

removeGroupFromRoleForEnvironment

Remove a group to a role for an environment

Format

```
serenara-client [global-args...] [global-flags...]
removeGroupFromRoleForEnvironment [args...]
```

Options

```
-group, --group
   Required. Name of the group

-role, --role
   Required. Name of the role

-application, --application
   Required. Name of the application

-environment, --environment
   Required. Name of the environment
```

removeGroupFromRoleForResource

Remove a group to a role for a resource

Format

```
serenara-client [global-args...] [global-flags...]
removeGroupFromRoleForResource [args...]
```

Options

```
-group, --group
   Required. Name of the group

-role, --role
   Required. Name of the role

-resource, --resource
   Required. Name of the resource
```

removeGroupFromRoleForUI

Remove a group to a role for the UI

Format

```
serenara-client [global-args...] [global-flags...]
removeGroupFromRoleForUI [args...]
```

Options

```
-group, --group
   Required. Name of the group

-role, --role
   Required. Name of the role
```

removeResourceFromGroup

Remove a resource from a resource group. Only works with static groups.

Format

```
serenara-client [global-args...] [global-flags...]
removeResourceFromGroup [args...]
```

Options

```
-resource, --resource
   Required. Name of the resource to remove

-group, --group
   Required. Path of the resource group to remove from
```

removeRoleForApplications

Create a role for applications

Format

```
serenara-client [global-args...] [global-flags...]
removeRoleForApplications [args...]
```

Options

```
-role, --role
   Required. Name of the role
```

removeRoleForComponents

Create a role for components

Format

```
serenara-client [global-args...] [global-flags...]
removeRoleForComponents [args...]
```

Options

```
-role, --role
   Required. Name of the role
```

removeRoleForEnvironments

Create a role for environments

Format

```
serenara-client [global-args...] [global-flags...]
removeRoleForEnvironments [args...]
```

Options

```
-role, --role
   Required. Name of the role
```

removeRoleForResources

Create a role for resources

Format

```
serenara-client [global-args...] [global-flags...]
removeRoleForResources [args...]
```

Options

```
-role, --role
   Required. Name of the role
```

removeRoleForUI

Create a role for the UI

Format

```
serenara-client [global-args...] [global-flags...]
removeRoleForUI [args...]
```

Options

```
-role, --role
   Required. Name of the role
```

removeRoleFromResource

Remove a role from a resource.

Format

```
serenara-client [global-args...] [global-flags...]
removeRoleFromResource [args...]
```

Options

```
-resource, --resource
   Required. Name of the parent resource.

-role, --role
   Required. Name of the new resource.
```

removeUserFromGroup

Remove a user from a group

Format

```
serenara-client [global-args...] [global-flags...]
removeUserFromGroup [args...]
```

Options

```
-user, --user
   Required. Name of the user

-group, --group
   Required. Name of the group
```

removeUserFromRoleForApplication

Remove a user to a role for an application

Format

```
serenara-client [global-args...] [global-flags...]
removeUserFromRoleForApplication [args...]
```

Options

```
-user, --user
   Required. Name of the user

-role, --role
   Required. Name of the role

-application, --application
   Required. Name of the application
```

removeUserFromRoleForComponent

Remove a user to a role for a component

Format

```
serenara-client [global-args...] [global-flags...]
removeUserFromRoleForComponent [args...]
```

Options

```
-user, --user
   Required. Name of the user

-role, --role
   Required. Name of the role

-component, --component
   Required. Name of the component
```

removeUserFromRoleForEnvironment

Remove a user to a role for an environment

Format

```
serenara-client [global-args...] [global-flags...]
removeUserFromRoleForEnvironment [args...]
```

Options

```
-user, --user
   Required. Name of the user

-role, --role
   Required. Name of the role

-application, --application
   Required. Name of the application

 -environment, --environment
   Required. Name of the environment
```

removeUserFromRoleForResource

Remove a user to a role for a resource

Format

```
serenara-client [global-args...] [global-flags...]
removeUserFromRoleForResource [args...]
```

Options

```
-user, --user
   Required. Name of the user
```

```
-role, --role
   Required. Name of the role

-resource, --resource
   Required. Name of the resource
```

removeUserFromRoleForUI

Remove a user to a role for the UI

Format

```
serenara-client [global-args...] [global-flags...]
removeUserFromRoleForUI [args...]
```

Options

```
-user, --user
   Required. Name of the user

-role, --role
   Required. Name of the role
```

repeatApplicationProcessRequest

Repeat an application process request.

Format

```
serenara-client [global-args...] [global-flags...]
repeatApplicationProcessRequest [args...]
```

Options

```
-request, --request
   Required. ID of the application process request to repeat
```

requestApplicationProcess

Submit an application process request to run immediately. This command takes a JSON request body. Use the -t flag to view the template for the data required for this command.

Format

```
serenara-client [global-args...] [global-flags...]
requestApplicationProcess [args...] [-] [filename]

-
   Read JSON input from the stdin. See command for requirements.
```

```
filename
   Read JSON input from a file with the given filename. See command for requirements.
```

Options

```
   No options for this command.
```

setComponentEnvironmentProperty

Set property on component/environment mapping

Format

```
serenara-client [global-args...] [global-flags...]
setComponentEnvironmentProperty [args...]
```

Options

```
-propName, --propName
   Required. Name of the property to set

-propValue, --propValue
   Required. Value of the property to set

-component, --component
   Required. Name of the component to look up

-environment, --environment
   Required. Name or id of the environment to look up

-application, --application
   Optional. Name of the application to look up
```

setComponentProperty

Set property on component

Format

```
serenara-client [global-args...] [global-flags...]
setComponentProperty [args...]
```

Options

```
-propName, --propName
   Required. Name of the property to set

-propValue, --propValue
   Required. Value of the property to set

-component, --component
   Required. Name of the component to look up
```

setResourceProperty

Set a custom property on a resource

Format

```
    serenara-client [global-args...] [global-flags...]
    setResourceProperty [args...]
```

Options

```
    -resource, --resource
       Required. Name of the resource to configure

    -name, --name
       Required. Name of the property

    -value, --value
       Optional. New value for the property
```

updateUser

Add a new user This command takes a JSON request body. Use the -t flag to view the template for the data required for this command.

Format

```
    serenara-client [global-args...] [global-flags...]
    updateUser [args...] [-] [filename]

    -
       Read JSON input from the stdin. See command for requirements.

    filename
       Read JSON input from a file with the given filename. See command for requirements.
```

Options

```
    -user, --user
       Required. Name of the user
```