Server Automation

Software Version: 10.50

Developer Guide

Document Release Date: July, 2016
Software Release Date: July, 2016

Hewlett Packard Enterprise

Legal Notices

Warranty

Restricted Rights Legend

Copyright Notice

Trademark Notices

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to: https://softwaresupport.hpe.com/.

This site requires that you register for an HPE Passport and to sign in. To register for an HPE Passport ID, click **Register** on the HPE Software Support site or click **Create an Account** on the HPE Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HPE sales representative for details.

Support

Visit the HPE Software Support site at: https://softwaresupport.hpe.com.

This website provides contact information and details about the products, services, and support that HPE Software offers.

HPE Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support website to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HPE support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HPE Passport user and to sign in. Many also require a support contract. To register for an HPE Passport ID, click **Register** on the HPE Support site or click **Create an Account** on the HPE Passport login page.

To find more information about access levels, go to: https://softwaresupport.hpe.com/web/softwaresupport/access-levels.

HPE Software Solutions Now accesses the HPESW Solution and Integration Portal website. This site enables you to explore HPE Product Solutions to meet your business needs, includes a full list of Integrations between HPE Products, as well as a listing of ITIL Processes. The URL for this website is https://softwaresupport.hpe.com/.

Contents

Developer guide overview

The Server Automation Platform is a set of APIs and a runtime environment that facilitate the integration and extension of SA. The Server Automation Platform APIs expose core services such as audit compliance, Windows patch management, and OS provisioning. The runtime environment executes Global Shell scripts that can access the Global File System (OGFS).

This topic provides information about the following:

Overview

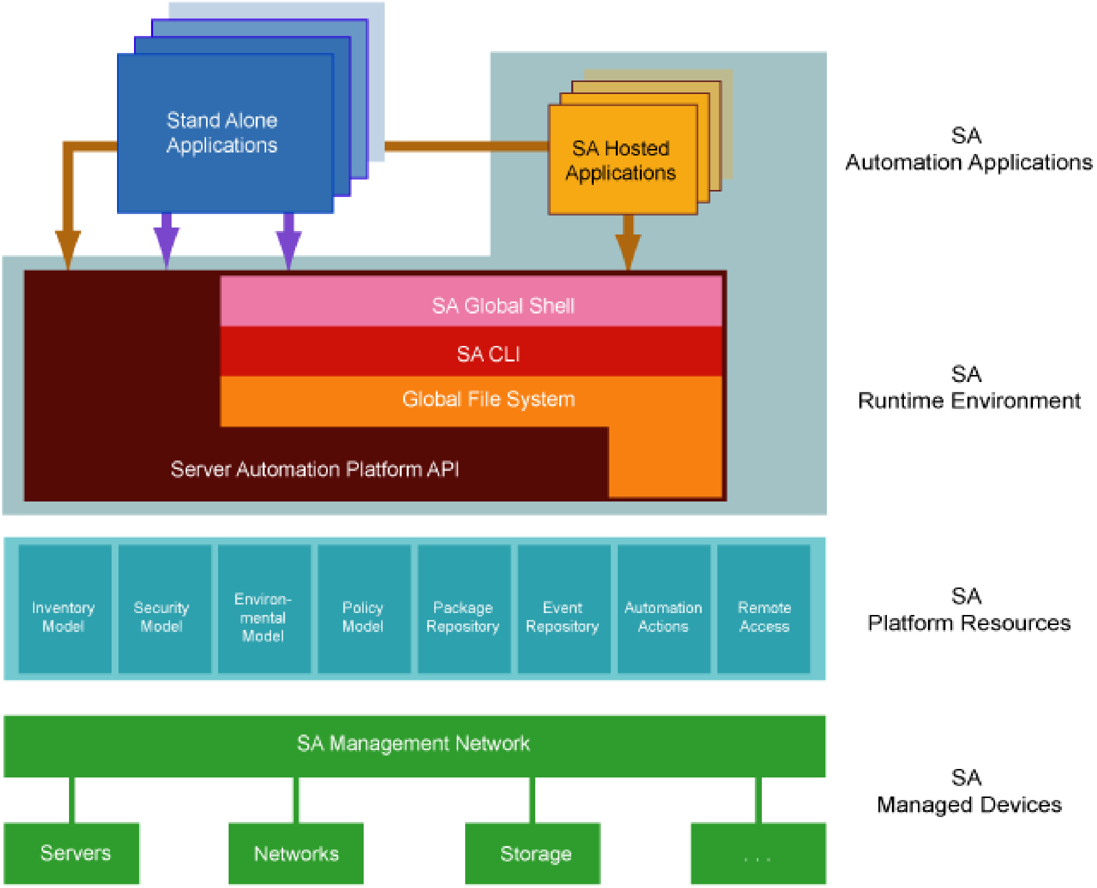
Using the Server Automation Platform, you can perform the following tasks:

- Build new automation applications and extend SA to improve IT productivity and comply with your IT policies.
- Exchange information with other IT systems, such as existing monitoring, trouble ticketing, billing, and virtualization technology.
- Use the SA Model Repository to store and organize critical IT information about operations, environment, and assets.
- Automate the management of a wide range of applications and operating systems.
- Incorporate existing Unix and Windows scripts with SA, enabling the scripts to run in a secure, audited environment.

Components

The following figure displays the major elements of the Server Automation Platform.

Server Automation Platform components



As the above figure shows, the platform comprises the following five key elements. Each of these elements is discussed in more detail in subsequent sections.

- **Automation applications:** The applications users write on top of the platform. These applications can either be SA-Hosted Applications which run in the context of the running SA or standalone applications running in the context of existing business and management systems.
- **Runtime environment:** Provides a set of powerful, out of the box runtime services and a corresponding language independent programming model explicitly designed to be easily accessibility to a broad spectrum of programmers, from script writers to Web developers to experienced enterprise Java programmers.
- **Platform resources:** Provide developers easy access to the platform’s rich data objects, automation actions (such as patching, provisioning, and auditing), and capabilities (such as remote access to each managed server’s runtime environment).
- **SA Management Network:** A powerful set of connectivity, security, and caching technologies which enable the platform to reach any device regardless of its location, IP address space, bandwidth availability, and so on.

- **SA Managed Devices:** The managed servers and network devices connected to the platform by the SA Management Network.

Automation applications

As in the figure above, the Automation Applications are at the top of the stack. These are the applications users write on top of the platform.

Automation applications can either be SA-Hosted Applications, which run in the SA Runtime Environment, or as standalone applications that run in a completely independent context. Standalone applications access the platform remotely through Web Services calls.

Simple applications can be written as simple Unix shell scripts in minutes. More complex applications—such as integration with an existing source control or ticketing system—can take a little longer and might involve Python or Microsoft .NET or Java coding. In either case, the platform is designed as a language-independent system easily adopted by a wide variety of developers.

SA runtime environment

Next down the platform stack is the SA Runtime Environment, which provides a set of powerful, out-of-the box runtime services and a corresponding language-independent programming model. SA-Hosted Applications run in the SA Runtime Environment.

The core of the runtime environment consists of two components: the Global Shell and the Global File System. Together, these two components organize and provide access to all managed devices in a familiar Linux/Unix shell file-and-directory paradigm.

Global Shell

The Global Shell is a command-line interface to the Global File System (OGFS). The command-line interface is exposed through a Linux shell such as `bash` that runs in a terminal window. The OGFS unifies the SA data model and the contents of managed servers—including files—into a single, virtual file system.

Global File System

The OGFS represents objects in the platform data model (such as facilities, customers, and device groups) and information available on platform managed devices (such as the configuration setting on a managed network device or the file system of a managed server) as a hierarchical structure of file directories and text files. For example, in the OGFS, the `/opsw/Customer` directory contains details about customer objects and the `/opsw/Server` directory has information about managed servers. The `/opsw/Server` directory also contains subdirectories that reflect the contents (such as file systems and registries) of the managed servers.

This file-and-directory paradigm allows administrators familiar with shell scripting to easily write scripts which perform the same task across different servers by iterating through the directories that represent servers. Behind the scenes, the Global File System securely delivers and executes any logic in the script to each managed server.

The contents of devices can be accessed through the Global File System, a virtual file system that represents all devices managed by SA and Network Automation (NA). Given the necessary security authorizations, both end users and automation applications can navigate through the OGFS to the file systems of remote servers. On Windows servers, administrators can also access the registry, II metabase, and COM+ objects.

SA Command Line Interface

The SA Command Line Interface (CLI) provides system administrators and platform automation applications a way to invoke automation tasks such as provisioning software, patching devices, or running audits from the command line. A rich syntax allows users to represent rich object types as input or receive them as output from CLI invocations.

The CLI itself is actually programmatically generated on top of the platform API, discussed in the next section. The advantage of this is that as soon as developers add a new API to the platform API, a corresponding CLI method is automatically available for it. In other words, there is no lag time between the availability of new features in the product and the availability of the corresponding CLI methods in the platform.

SA Platform API

The SA Platform API is the Win32 API of SA: It defines a set of application programming interfaces to get and set values as well as perform actions. The SA user interfaces, including the SA Client and the SA Command Line Interfaces (CLI), are all built on top of the SA Platform API. The API includes libraries for Java RMI clients and WSDLs for SOAP-based Web Services clients. With Web Services support, programmers can create clients in popular languages such as Perl, C#, and Python.

SA Platform resources

SA Platform Resources sit beneath the SA Runtime Environment and give developers access to a rich set of objects and actions which they can re-use and manipulate in their own applications.

Inventory Model

The Inventory Model provides all the information gathered by the SA about each managed devices such as make, manufacturer, CPU, operating system, installed software, and so on. Inventory information is made available through the SA API and also appears as files (in the `attr` subdirectories) in the Global File System. The Inventory Model includes objects such as Servers and Network Devices.

Administrators can extend the data associated with inventory objects. For example, if users want to store a picture of the device or a lease expiration date or the ID of a UPS the device is plugged into, the platform makes it easy to add those attributes to each device record. Users can then add, delete, and work with those attributes just as they would the attributes that come out of the box.

Security Model

The Security Model allows developers to leverage the built-in SA authentication and authorization security systems.

All clients of the platform—management applications, scripts, as well as the end-user interfaces provided by SA are controlled by the same security framework.

The security administrator — not the developer — creates user roles and grants permissions. Developers can re-use all of these user roles and permissions in the context of their own applications. For example, network administrators can write a shell script and share it with other network administrators with the confidence that those network administrators can only run that script on network devices they are authorized to manage and no others.

The authorization mechanism controls access at several levels: the types of tasks users can perform, the servers and network devices accessed by the tasks, and the SA objects (such as software policies).

Environment Model

The Environment Model defines the overall business context in which devices live. In general, devices belong to one or more customers, are located in a particular facility, and belong to one or more groups. The platform makes each of these objects — Customers Facilities, Device Groups, and others — available to application developers.

As with inventory objects, environment objects can easily be extended. This makes it easy, for example, to define attributes such as the SNMP trap receiver used in a particular data center or printers only available in a particular facility, or Apache configurations used by only a particular business unit.

Policy Model

The Policy Model gives developers access to all the best practices defined in SA. Policies describe the desired state on a server or network device. For example, a patch policy describes the patches that should be on a server, a software policy describes what software should be on a server, and so on.

Subject matter experts define these policies which can be used by any authorized system administrator to audit devices to discover whether what’s actually on a device differs from what should be on the device. Programmers have access to this complete library of policies to use in their own applications.

Software policies are organized into folders which can define security boundaries. In other words, applications will be able to access only those software policies they are permitted to access based on their user permissions.

Package Repository

The Package Repository gives developers access to all the software and patches stored in SA. These include operating system builds, operating system patches, middleware, agents, and any other pieces of software that users have uploaded into SA.

Event Repository

The Event Repository houses the digitally signed audit trails that the SA generates when actions are performed, either through the user interface or programmatically with the platform. As with other platform objects, these events are available programmatically.

Automation Actions

Automation Actions allow developers to programmatically launch any of the actions that SA can perform on managed devices, ranging from running an audit to provisioning software to applying the latest OS patch.

The platform provides access to the same features available to end-users in the SA Client. These features include tasks such as installing patches, provisioning operating systems, and installing and removing software policies. In fact, the SA Client calls the same APIs that are exposed programmatically through the SA Runtime Environment.

Remote Access

Remote Access gives developers programmatic access to the managed device’s file system (in the case of servers) and execution environment (in the case of all devices). Developers can easily write applications which check for the existence of a file or particular software package, run operating system commands to check disk usage, or run system scripts to perform routine maintenance tasks.

SA Management Network

The Management Network is a powerful combination of technologies which enable developers to securely access any device under management. The Management Network delivers several key services:

- **Connectivity**: Allows the platform (and thus automation applications) to reach any managed device.
- **Security**: Includes SSL/TLS-based encryption, authentication, and message integrity.
- **Address space virtualization**: Enables the platform to locate servers across multiple overlapping IP address spaces. Most complex enterprise networks have multiple private IP address spaces.
- **Availability**: Allows system architectures to define redundant paths to any given managed device so that devices can still be reached despite failures in any given network path.
- **Caching**: Enables servers to download software and patches from a nearby server rather than a distant server, saving both time and network connectivity charges.
- **Bandwidth throttling**: Lets system architectures determine how much bandwidth SA and any SA applications can consume as it traverses the network to a particular device.
- **Least cost routing**: Allows system designers to set up rules governing which paths to use to reach a particular device to minimize network connectivity costs.

SA Managed Devices

At the bottom of the platform stack are the actual devices under management. The platform manages over 65 server OS versions and over 35 different network device vendors with thousands of device models/versions supported out of the box.

The list of supported devices is constantly being updated. Platform developers and script writers benefit directly from this device list since their automation applications can consistently reach an ever growing list of managed devices in the same, familiar platform programming environment.

Benefits of the SA Platform

The SA Platform has the following key benefits.

Powerful security

The platform delivers the following comprehensive security mechanisms so developers don’t have to worry about providing them in their own applications.

- **Secure communication channels**: End-to-end communication from the automation applications out to the managed devices is encrypted and authenticated.
- **Role-based access control**: The platform respects the role-based access controls built into the SA so developers can easily share their applications with the con.dence that they will run just on those devices that an administrator has been granted access to.
- **Digitally signed audit trail**: After an automation application runs, the platform generates a digitally signed audit trail capturing who ran the application, the time of the application execution, and the devices on which the application ran.
- **Comprehensive reach** The platform provides comprehensive reach across all devices so system administrators and developers don’t have to worry about how to get to a device:
- **Market-leading platform coverage**: Supported devices include over 65 server OS versions and more than 1,000 network devices.
- **In any physical location**: The devices can be located anywhere in the world whether in a major data center or a retail store or a satellite of.ce.
- **In any IP address space**: The devices can belong to any IP address space, as the platform supports multiple overlapping IP address spaces.
- **In DMZs**: Devices can be located in DMZs or other difficult-to-access network spaces without requiring the developer or system administrator to worry about the details of reaching the device (for example, through a bastion host).

Rich services

The platform exposes practically all the relevant data and actions in the underlying automation system:

- **Rich data out-of-the-box**: Developers have easy access to a rich set of data generated in part by the platform itself (such as device inventory data and facility information) and in part by users interacting with the platform (such as device groups customers, best practices policies, and uploaded software, patches, and scripts). Developers can easily write applications to read and write this data.
- **Extensible data store**: Developers can easily extend the native platform objects to include their own data. Device inventory models can be extended to include attributes the platform does not natively discover. Customer and facility objects can be extended to include attributes that should guide the provisioning or auditing of devices related to that customer.
- **Automation tasks**: The platform exposes nearly all the capabilities of the underlying automation systems to developers: patching, provisioning, auditing, and others. This enables developers

writing complex work flows that span multiple systems to simply call these actions from the context of an automation application.

Easily accessible to a broad spectrum of programmers

The platform is explicitly designed to appeal to a broad range of developers ranging from Unix shell and Visual Basic script writers to Perl and Python programmers to enterprise .NET or Java programmers. The platform’s Runtime Services layer makes most platform objects available in a file-and-directory paradigm and most platform services available from a command-line interface (the SA CLI). This allows system administrators used to writing shell scripts to instantly use the platform without having to learn a new programming language and tool. They can get started with their favorite text editor, a familiar Unix shell, and then quickly develop scripts.

For more complicated applications and integration with existing systems, system programmers can use whatever programming tools and languages that have Web Services bindings.

SA Platform API design

The Platform API is defined by Java interfaces and organized into Java packages. To support a variety of client languages and remote access protocols, the API follows a function-oriented, call-by-value model.

Services

In the Platform API, a service encapsulates a set of related functions. Each service is specified by a Java interface with a name ending in `Service`, such as `ServerService`, `FolderService`, and `JobService`.

Services are the entry points into the API. To access the API, clients invoke the methods defined by the server interface. For example, to retrieve a list of software installed on a managed server, a client invokes the `getInstalledSoftware` method of the `ServerService` interface. Examples of other `ServerService` methods are `checkDuplex`, `setPrimaryInterface`, and `changeCustomer`.

The SA Platform API contains over 70 services – too many to describe here. The following table lists a few of the services that you may want to try out first. For a full list of services, in a browser go to the URL shown in "API Documentation and the Twister" on page 37.

Partial list of services of the SA API

Service name	Some of the operations provided by this service
AuditTaskService	Create, get, and run audit tasks.
ConfigurationService	Create application configurations, get the software policies using an application configuration.
DeviceGroupService	Create device groups, assign devices to groups, get members of groups, set dynamic rules.
EventCacheService	Trigger actions such as updating a client-side cache of value objects. See "Event Cache" on page 35.
FolderService	Create folders, get children of folders, set customers of folders, move folders.
InstallProfileService	Create, get, and update OS installation profiles.
JobService	Get progress and results of jobs, cancel jobs, update job schedules.

Partial list of services of the SA API, continued

Service name	Some of the operations provided by this service
NasConnectionService	Get host names of NA servers, run commands on NA servers.
NetworkDeviceService	Get information such as families, names, models, and types, according to specified search filters.
SequenceService	Create, get, and run OS sequences to install operating systems on servers.
ServerService	Get information about servers, reconcile (remediate) policies on servers (install software), get and set custom fields and attributes, execute OS sequences (install OS).
SoftwarePolicyService	Create software policies, assign policies to servers, get contents of policies, remediate (reconcile) policies with servers.
SolPatchService	Install and uninstall Solaris patches, add policy overrides.
VirtualColumnService	Manage custom fields and custom attributes.
WindowsPatchService	Install and uninstall Windows patches, add policy overrides.

Objects in the API

Although the SA Platform API is function-oriented, its design enables clients to create object-oriented libraries. TheSA data model includes objects such as servers, folders, and customers. These are persistent objects; that is, they are stored in the Model Repository. In the API, these objects have the following items:

- A service that defines the object’s behavior. For example, the methods of the `ServerService` specify the behavior of a managed server object.
- An object (identity) reference that represents an instance of a persistent object. For example, `ServerRef` is a reference that uniquely identifies a managed server. In the `ServerService`, the first parameter of most methods is `ServerRef`, which identifies the managed server operated on by the method. The `Id` attribute of a `ServerRef` is the primary key of the server object stored in the Model Repository.
- One or more value objects (VOs) that represent the data members (attributes, fields) of a persistent object. For example, `ServerVO` contains attributes such as `agentVersion` and `loopbackIP`. The attributes of `ServerHardwareVO` include `manufacturer`, `model`, and `assetTag`. Most attributes cannot be changed by client applications. If an attribute can be changed, then the API documentation for the setter method includes “Field can be set by clients.”

For performance reasons, update operations on persistent objects are coarse-grained. The `update` method of `ServerService`, for example, accepts the entire `ServerVO` as an argument, not individual attributes.

Exceptions

All of the API exceptions that are specific to SA are derived from one of the following exceptions:

`OpswareException` - Thrown when an application-level error occurs, such as when an end-user enters an illegal value that is passed along to a method. Typically, the client application can recover from this type of exception. Examples of exceptions derived from `OpswareException` are `NotFoundException`, `NotInFolderException`, and `JobNotScheduledException`.

`OpswareSystemException` - Thrown when an error occurs within SA. Usually, the SA Administrator must resolve the problem before the client application can run.

The following exceptions are related to security:

`AuthenticationException` - Thrown when an invalid SA user name or password is specified.

`AuthorizationException` - Thrown when the user does not have permission to perform an operation or access an object. For more information on permissions, see the SA 10.50 Administration Guide.

Event Cache

Some client applications need to keep local copies of SA objects. Accessed by clients through the `EventCacheService`, the cache contains events that describe the most recent change made to SA objects. Clients can periodically poll the cache to check whether objects have been created, updated, or deleted. The cache maintains events over a configured sliding window of time. By default, events for the most recent two hours are maintained. To change the sliding window size, edit the Web Services Data Access Engine configuration file, as described in the Server Automation Administration Guide on the HPE SSO portal.

Searches

The search mechanism of the SA Platform API retrieves object references according to the attributes (fields) of value objects. For example, the `getServerRefs` method searches by attributes of the `ServerVO` value object. The `getServerRefs` method has the following signature:

```
public ServerRef[] getServerRefs(Filter filter)...
```

Each `get*Refs` method accepts the `filter` parameter, an object that specifies the search criteria. A `filter` parameter with a simple expression has the following syntax:

value-object.attribute operator value

(This syntax is simplified. For the full definition, see "Filter grammar" on page 216.)

The following examples are `filter` parameters for the `getServerRefs` method:

```
ServerVO.hostName = "d04.example.com"
ServerVO.model BEGINS_WITH "POWER"
ServerVO.use IN "UNKNOWN" "PRODUCTION"
```

Complex expressions are allowed, for example:

```
(ServerVO.model BEGINS_WITH "POWER") AND (ServerVO.use = "UNKNOWN")
```

Not every attribute of a value object can be specified in a `filter` parameter. For example, `ServerVO.state` is allowed in a `filter` parameter, but `ServerVO.OsFlavor` is not. To find out which attributes are allowed, locate the value object in the API documentation and look for the comment, “Field can be used in a filter query.”

Security

Users of the SA Platform must be authenticated and authorized to invoke methods on the SA Automation Platform API. To connect to SA, a client supplies an SA user name and password (authentication). To invoke methods, the SA user must belong to a user group with the necessary permissions (authorization). These permissions restrict not only the types of operations that users can perform, but also limit access to the servers and network devices used in the operations.

Before application clients can run on the platform, the SA Administrator must specify the required users and permissions with the Command Center. For instructions, see the "User Group and Setup" section in the SA 10.50 Administration Guide. For information about security-related exceptions, see "Exceptions" on the previous page.

Communication between clients and SA is encrypted. For Web Services clients, the request and response SOAP messages (which implement the operation calls) are encrypted using SSL over HTTP (HTTPS).

API Documentation and the Twister

SA includes API documentation (Javadocs) that describe the SA Platform API. To access the API documentation, specify the following URL in a browser:

```
https://<SA_core_host>/twister
```

The <*SA_core_host*> is the IP address or host name of the SA core server running the Command Center component.

The *Twister* is a program that lets you invoke API methods, one at a time, from within a browser. For example, to invoke the `ServerService.getServerVO` method, perform the following steps:

1. Open the API documentation in a browser.
2. In the All Classes pane, select `com.opsware.server`.
3. In the `com.opsware.server` pane, select `ServerService`.
4. In the main pane, scroll down to the `getServerVO` method.
5. Click **Try It** for the `getServerVO` method.
6. Enter your SA user name and password.
7. In the Twister pane for `ServerService.getServerVO`, enter the ID of a managed server in the `oid` field.
8. Click **Go**. The Twister pane displays the attributes of the `ServerVO` object returned.

Constant field values

Some of the API's value objects (VOs) have fields with values defined as constants. For example, `JobInfoVO` has a `status` field that can have a value defined by constants such as `STATUS_ACTIVE`, `STATUS_PENDING`, and so forth. The API specifies constants as Java `static final` fields, but the WSDLs generated from the API do not define the constants. To view the definitions for constants, in the API documentation, go to the Constant Field Values page:

```
https://<SA_core_host>/twister/docs/constant-values.html
```

For example, the Constant Field Values page defines `STATUS_ACTIVE` as the integer 1.

Supported clients

The SA platform supports programmers with different skills, from system administrators who write shell scripts to .NET and Java programmers familiar with the latest tools and technologies. All supported clients call the same set of methods, which are organized into the services of the SA Platform. A developer can create the following types of clients that call methods in the SA Platform API:

- **SA Command-Line Interface (CLI)**: Launched from Global Shell sessions, shell scripts can access the SA Platform API by invoking the CLI methods, which are executable programs in the OGFS. Each CLI method corresponds to a method in the API.
- **Web Services**: Using SOAP over HTTPS, these clients send requests to SA and get responses back. The Web Services operations (defined in WSDLs) correspond to the methods in the API. Developers can write Web Services clients in popular languages such as Perl and C#.
- **Java RMI**: These clients invoke remote Java objects from other Java virtual machines.
- **Pytwist**: These Python programs can run on an SA Core or managed servers.

The Web Services and Java RMI clients can run on servers different than the SA Core or managed servers. The CLI methods execute in a Global Shell session on the core server where the OGFS is installed.

SA CLI methods

End-users access SA through the SA Client. At times, advanced users need to access SA in a command-line environment to perform bulk operations or repetitive tasks on multiple servers. In SA, the command-line environment consists of the Global Shell (OGSH), Global File System (OGFS), and SA Command-Line Interface (CLI) methods.

To perform SA operations from the command line, you invoke the SA CLI methods from within an OGSH session. An SA CLI method is an executable in the OGFS that corresponds to a method in the SA API. When you run an SA CLI method, the underlying API method is invoked.

To understand this section, you should be familiar with the OGSH and the OGFS. For more information, see the OGSH in the Server Automation Using Guide on the HPE SSO portal.

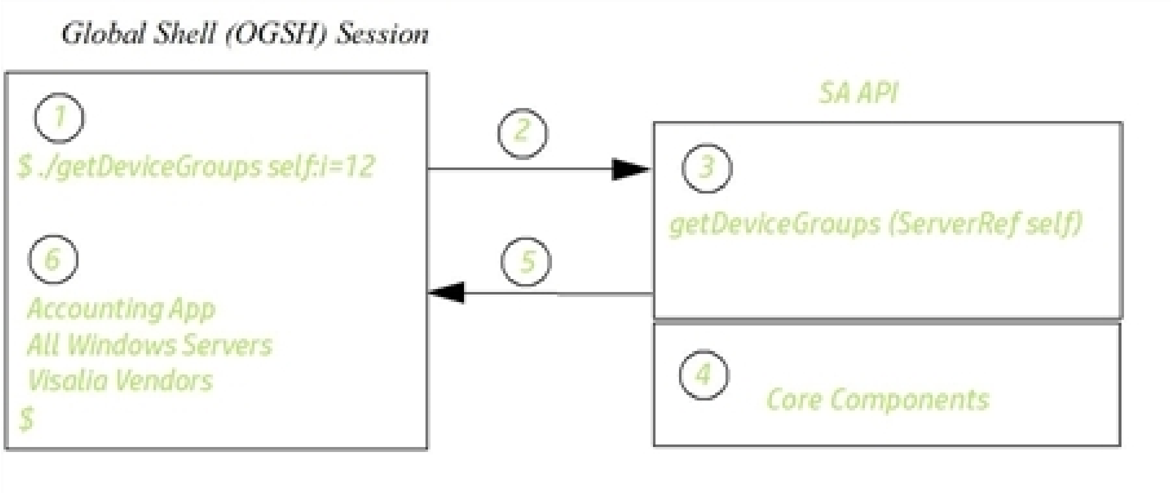
For information on the `oupload` and `odownload` commands, see the OCLI 1.0 in the Server Automation Using Guide on the HPE SSO portal.

Method invocation

As shown in the following figure, when you invoke an SA CLI method in an OGSH session, the following operations occur:

1. The OGSH parses the command and parameters you entered to determine the API method.
2. The OGSH invokes the underlying API method.
3. An authorization check verifies that the user has permission to perform this operation. SA then performs the operation.
4. The API method passes the results back to the SA CLI method.
5. The SA CLI method writes the return value to the `stdout` of the OGSH session. If an exception was thrown, the SA CLI method returns a non-zero status.

Overview of an SA CLI method invocation


Global Shell (OGSH) Session
1
$ ./getDeviceGroups self:i=12
6
Accounting App
All Windows Servers
Visalia Vendors
$
2
5
SA API
3
getDeviceGroups (ServerRef self)
4
Core Components

Security

SA CLI methods use the same authentication and authorization mechanisms as the SA Client. When you start an OGSH session, SA authenticates your SA user. When you run an SA CLI method, authorization is performed. To run an SA CLI method successfully, your SA user must belong to a group that has the required permissions. For more information on security, see the Server Automation Administration Guide on the HPE SSO portal.

Mapping between API and SA CLI methods

The OGFS represents SA objects as directory structures, object attributes as text files, and API methods as executables. These executables are the SA CLI methods. Every SA CLI method matches an underlying API method. The method name, parameters, and return value are the same for both types of methods.

For example, the `setCustomer` API method has the following Java signature:

```
public void setCustomer(ServerRef self,

          CustomerRef customer)...
```

In the OGFS, the corresponding SA CLI method has the following syntax:

```
setCustomer self:i=server-id customer:i=customer-id
```

Note that the parameter names, `self` and `customer`, are the same in both languages. (The `:i` notations are called format specifiers, which are discussed later in this section.) In this example, the return type is void, so the SA CLI method does not write the result to the `stdout`. For information on how SA CLI methods return strings that represent objects, see "Return values " on page 59.

Differences between SA CLI methods and Unix commands

Although you can run both Unix commands and SA CLI methods in the OGSH, SA CLI methods differ in several ways:

- Unlike many Unix commands, SA CLI methods do not read data from `stdin`. Therefore, you cannot insert an SA CLI method within a group of commands connected by pipes (|). (However, SA CLI methods do write to `stdout`.)
- Most Unix commands accept parameters as flags and values (for example, ls `-l /usr`). With SA CLI methods, command-line parameters are name-value pairs, joined by equal signs.
- Unix commands are text based: They accept and return data as strings. In contrast, SA CLI methods can accept and return complex objects.
- With SA CLI methods, you can specify the format of the parameter and return values. Unix commands do not have an equivalent feature.

SA CLI method tutorial

This topic introduces you to the SA CLI methods with examples you can try in your own environment. After completing this tutorial, you should be able to run SA CLI methods, examine the `self` file of an SA object, and create a script that invokes SA CLI methods on multiple servers.

Before starting the tutorial, you need the following capabilities:

- You can log on to the SA Client.
- Your SA user has Read & Write permissions on at least one managed server. Typically assigned by a security administrator, permissions are discussed in the Server Automation Administration Guide on the HPE SSO portal.
- Your SA user has all OGSH permissions on the same managed server. For information on these permissions, see the “aaa Utility” section in the Server Automation Using Guide on the HPE SSO portal.

- You are familiar with the OGSH and the OGFS. If these features are new to you, before proceeding with this tutorial, see the Global Shell section inthe Server Automation Using Guide on the HPE SSO portal.

The example commands in this tutorial operate on a Windows server named `abc.example.com`. This server belongs to a server group named All Windows Servers. When trying out these commands, substitute `abc.example.com` with the host name of the managed server you have permission to access.

1. Open an OGSH session.

   You can open a Global Shell session from within the SA Client. From the **Actions** menu, select **Global Shell**. You can also open an OGSH session from a terminal client running on your desktop. For instructions, see Opening a Global Shell Session section in the Server Automation Using Guide on the HPE SSO portal.

2. List the SA CLI methods for a server.

   The `method` subdirectory of a specific server contains executable files—the methods you can run for that server. The following example lists the SA CLI methods for the `abc.example.com` server:

   ```
   $ cd /opsw/Server/@/abc.example.com/method
   $ ls -1
   addDeviceGroups
   attachPolicies
   attachVirtualColumn
   checkDuplex
   clearCustAttrs
   ...
   ```

   These methods have instance context – they act on a specific server instance (in this case, `abc.example.com`). The server instance can be inferred from the path of the method. Methods with static context are discussed in step 5.

3. Run an SA CLI method without parameters.

   To display the public server groups that `abc.example.com` belongs to, invoke the `getDeviceGroups` method:

   ```
   $ cd /opsw/Server/@/abc.example.com/method
   $ ./getDeviceGroups
   ```

```
Accounting App
All Windows Servers
Visalia Vendors
```

4. Run a method with a parameter.

   Command-line parameters for methods are indicated by name-value pairs, separated by white space characters. In the following invocation of `setCustomer`, the parameter name is `customer` and the value is 20039. The `:i` at the end of the parameter name is an ID format specifier, which is discussed in a later step.

   The following method invocation changes the customer of the `abc.example.com` server from Opsware to C39. The ID of customer C39 is 20039.

   ```
   $ cd /opsw/Server/@/abc.example.com
   $ cat attr/customer ; echo
   Opsware
   $ method/setCustomer customer:i=20039
   $ cat attr/customer ; echo
   C39
   ```

5. List the static context methods for managed servers.

   Static context methods reside under the `/opsw/api` directory. These methods are not limited to a specific instance of an object.

   To list the static methods for servers, enter the following commands:

   ```
   $ cd /opsw/api/com/opsware/server/ServerService/method

   $ ls
   ```

   The methods listed are the same as those displayed in step 2.

6. Run a method with the `self` parameter.

   This step invokes `getDeviceGroups` as a static context method. Unlike the instance context method shown in step 3, the static context method requires the `self` parameter to identify the server instance.
   For example, suppose that the `abc.example.com` server has an ID of 530039. To list the groups of this server, enter the following commands:

```
$ cd /opsw/api/com/opsware/server/ServerService/method
$ ./getDeviceGroups self:i=530039
Accounting App
All Windows Servers
Visalia Vendors
```

Compare this invocation of `getDeviceGroups` with the invocation in step 3 that demonstrates instance context. Both invocations run the same underlying method in the API and return the same results.

7. Examine the `self` file of a server.

   Within SA, each managed server is an object. However, OGFS is a file system, not an object model. The `self` file provides access to various representations of an SA object. These representations are the ID, name, and structure.

   The default representation for a server is its name. For example, to display the name of a server, enter the following commands:

   ```
   $ cd /opsw/Server/@/abc.example.com
   $ cat self ; echo
   abc.example.com
   ```

   If you know the ID of a server, you can get the name from the `self` file, as in the following example:

   ```
   $ cat /opsw/.Server.ID/530039/self ; echo
   abc.example.com
   ```

8. Indicate an ID format specifier on a `self` file.

   To select a particular representation of the `self` file, enter a period, then the file name, followed by the format specifier. For example, the following `cat` command includes the format specifier (`:i`) to display the server ID:

   ```
   $ cd /opsw/Server/@/abc.example.com
   $ cat .self:i ; echo
   com.opsware.server.ServerRef:530039
   ```

This output shows that the ID of `abc.example.com` is 530039. The `com.opsware.server.ServerRef` is the class name of a server reference, the corresponding object in the SA API.

> **Note:** The leading period is required with format specifiers on files and method return values, but is not indicated with method parameters.

9. Indicate the structure format specifier.

   The structure format specifier (`:s`) indicates the attributes of a complex object. The attributes are displayed as name-value pairs, all enclosed in curly braces. Structure formats are used to specify method parameters on the command-line that are complex objects. For an example method call, see "Complex objects and arrays as parameters" on page 58.

   The following example displays `abc.example.com` with the structure format:

   ```
   $ cd /opsw/Server/@/abc.example.com
   $ cat .self:s ; echo
   {
   managementIP="192.168.8.217"
   modifiedBy="spujare"
   manufacturer="DELL COMPUTER CORPORATION"
   use="UNKNOWN"
   discoveredDate=1149012848000
   origin="ASSIMILATED"
   osSPVersion="SP4"
   locale="English_United States.1252"
   reporting=false
   netBIOSName=
   previousSWReg=1150673874000
   osFlavor="Windows 2000 Advanced Server"
   . . .
   ```

   The attributes of a server are also represented by the files in the `attr` directory, for example:

   ```
   $ pwd
   /opsw/Server/@/abc.example.com
   $ cat attr/osFlavor ; echo
   Windows 2000 Advanced Server
   ```

10. Create a script that invokes an SA CLI method.

    The example script shown in this step iterates through the servers of the public server group named All Windows Servers. On each server, the script runs the `getCommCheckTime` SA CLI method.

    First, return to your home directory in the OGFS:

    ```
    $ cd
    $ cd public/bin
    ```

    Next, run the `vi` editor:

    ```
    $ vi
    ```

    In `vi`, insert the following lines to create a `bash` script:

    ```
    #!/bin/bash
    # iterate_time.sh

    METHOD_DIR="/opsw/api/com/opsware/server/ServerService/method"
    GROUP_NAME="All Windows Servers"
    cd "/opsw/Group/Public/$GROUP_NAME/@/Server"

    for SERVER_NAME in *
    do
     SERVER_ID=`cat $SERVER_NAME/.self:i`
     echo $SERVER_NAME
     $METHOD_DIR/getCommCheckTime self:i=$SERVER_ID
     echo
     echo
    done
    ```

    Save the file in `vi`, naming it iterate_time.sh. Quit `vi`.

    Change the permissions of iterate_time.sh with chmod, and then run it:

    ```
    $ chmod 755 iterate_time.sh
    $ ./iterate_time.sh
    abc.example.com
    ```

```
2006/06/20 16:46:56.000
. . .
```

Format specifiers

Format specifiers indicate how values are displayed or interpreted in the SA CLI environment. You can apply a format specifier to a method parameter, a method return type, the `self` file, and an object attribute. To indicate a format specifier, append a colon followed by one of the letters shown in the following table.

If a format specifier is indicated for a file or a method return value, a period must precede the file or method name. For method return values that have format specifiers, the leading period is not included.

Summary of format specifiers

Format specifier	Description	Valid object types	Allowed as method parameter?
:n	**Name**: A string identifying the object. Unique names are preferred, but not required. For objects that do not have a name, this representation is the same as the ID representation.	SA objects	Yes. If the name is ambiguous, an error occurs.
:i	**ID**: A format that uniquely identifies the object type and its SA ID. Also known as an object reference.	SA objects; Dates (`java.util.Calendar`) objects	Yes. If the type is clear from the context, the type may be omitted.
:s	**Structure**: A compact representation intended for specifying complex values on the command-line. Attributes are enclosed in curly braces.	Any complex object	Yes
:d	**Directory**: Represents an attribute as a directory in the OGFS.	Any complex object that is an attribute. This representation cannot be used for method parameters or return values.	No

Position of format specifiers

A format specifier immediately follows the item it affects. For files, a format specifier follows the file name. In the following example, note the leading period:

```
cat .self:s
```

When applied to a method return type, a format specifier follows the method name. The following invocation displays the IDs of the groups returned:

```
./.getDeviceGroups:i
```

With method parameters, a format specifier follows the parameter name and precedes the equal sign, as in the following example:

```
./setCustomer self:i=9977 customer:i=239
```

A method parameter with a format specifier does not have a leading period.

Default format specifiers

Every value or object has a default format specifier. For example, the name format specifier is the default for the `osVersion` attribute. The following two `cat` commands generate the same output:

```
cd /opsw/Server/@/d04.example.com/attr
cat osVersion
cat .osVersion:n
```

The name format specifier is the default for SA objects stored in the Model Repository, such as servers and customers. The structure format specifier is the default for other complex objects.

Examples of ID format specifier

The next example displays the ID of the facility that the `d04.example.com` server belongs to:

```
cd /opsw/Server/@/d04.example.com/attr
cat .facility:i ; echo
```

(The preceding `echo` command is optional. It generates a new-line character, which makes the output easier to read. The semicolon separates `bash` statements entered on the same line.)

The output of a value with the ID format specifier is prefixed by the Java class name. For example, if the facility value has an ID of 39, then the previous `cat` command displays the following output:

```
com.opsware.locality.FacilityRef:39
```

The following invocation of the `getDeviceGroups` method lists the IDs of the public server groups that `d04.example.com` belongs to:

```
cd /opsw/Server/@/d04.example.com/method
./.getDeviceGroups:i
```

For more ID format examples, see "The self file" on page 54.

Structure format specifier syntax

The structure format represents complex objects, which can contain various attributes. You might use this format to specify a method parameter that is a complex object. For examples, see "Complex objects and arrays as parameters" on page 58.

The structure format is a series of name-value pairs, separated by white space characters, enclosed in curly braces. Each name-value pair represents an attribute. The structure format has the following syntax:

```
{ name-1=value-1 name-2=value-2 . . . }
```

Here’s a simple example:

```
{ version=10.1.3 isCurrent=true }
```

Any white space character can be used as a delimiter:

```
{
       version=10.1.3
       isCurrent=true
}
```

Attributes can be specified as structures, enabling the representation of nested objects. In the following example, the `versionDesc` attribute is represented as a structure:

```
{
program=agent
versionDesc={
               version=10.1.3
               isCurrent=true
               comment="Latest version"
               }
}
```

To specify an array within a structure, repeat the attribute name. The following structure contains an array named `steps` that has three elements with the values 33, 14, and 28.

```
{ moduleName="Some Initiator" steps=33 steps=14 steps=28 }
```

Examples of structure format specifier

The following example specifies the structure format for the `facility` attribute:

```
cd /opsw/Server/@/d04.example.com/attr
cat .facility:s
```

This `cat` command generates the following output. Note that `customers` is an array, which contains an element for every customer associated with this facility.

```
{
       modifiedBy="192.168.9.246"
       customers="Customer Independent"
       customers="Not Assigned"
       customers="Opsware Inc."
       customers="Acme Inc."
       . . .
       ontogeny="PROD"
       createdBy=
       status="ACTIVE"
       createdDt=-1
       realms="Transitional"
       realms="C39"
       realms="C39-agents"
       modifiedDt=1146528752000
       name="C39"
       displayName="C39"
}
```

The following invocation of `getDeviceGroups` indicates the structure format specifier for the return value:

```
cd /opsw/Server/@/d04.example.com/method
./.getDeviceGroups:s
```

This call to `getDeviceGroups` displays the following output. Because `d04.example.com` belongs to two server groups, the output includes two structures. In each structure, the `devices` array has elements for the servers belonging to that group.

```
{
       dynamic=true
       devices="m302-w2k-vm1.dev.example.com"
       devices="d04.example.com"
       . . .
       status="ACTIVE"
       34 Chapter 2
       public=true
       fullName="Device Groups Public All Windows Servers"
       description="test"
       createdDt=-1
       modifiedDt=1142019861000
       parent="Public"
}
{
       dynamic=true
       devices="opsware-nibwp.build.example.com"
       devices="glengarriff.snv1.dev.example.com"
       devices="millstreet"
       . . .
       fullName="Device Groups Public z_testsrvgroup"
       . . .
}
```

The structure format specifier is the default for methods that retrieve value objects (VOs). For example, the following two calls to `getServerVO` are equivalent:

cd /opsw/Server/@/d04.example.com/method
./.getServerVO:s./getServerVO

In this example, `getServerVO` displays the following output:

```
{
       managementIP="192.168.198.93"
       modifiedBy=
       manufacturer="DELL COMPUTER CORPORATION"
       use="UNKNOWN"
       discoveredDate=1145308867000
       origin="ASSIMILATED"
       osSPVersion="RTM"
       locale="English_United States.1252"
       reporting=false
       netBIOSName=
       previousSWReg=1147678609000
       osFlavor="Windows Server 2003, Standard Edition"
       peerIP="192.168.198.93"
       modifiedDt=1145308868000
       . . .
       serialNumber="HVKZS51"
}
```

This structure represents the `ServerVO` class of the SA API. Every attribute in this structure corresponds to a file in the `attr` directory. In the next example, the `getServerVO` and `cat` commands both display the value of the `serialNumber` attribute of a server:

```
cd /opsw/Server/@/d04.example.com
./method/getServerVO | grep serialNumber
cat attr/serialNumber ; echo
```

Examples of directory format specifier

The following command changes the current working directory to the customer associated with the server `d04.example.com`:

```
cd /opsw/Server/@/d04.example.com/attr/.customer:d
```

The next command lists the name of this customer:

```
cat /opsw/Server/@/d04.example.com/attr/\
.customer:d/attr/name
```

The directory specifier can be used only in command arguments that require directory names. The following `cat` command fails because it attempts to display a directory:

```
cat /opsw/Server/@/d04.example.com/attr/.customer:d # WRONG!
```

However, the next command is legal:

```
ls /opsw/Server/@/d04.example.com/attr/.customer:d
```

Value representation

Because they run in a shell environment (the OGSH), SA CLI methods accept and return data as strings. However, the underlying API methods can accept and return other data types, such as numbers, Booleans, and objects. The sections that follow describe how the OGFS and SA CLI methods represent non-string data types.

SA Objects in the OGFS

The SA data model includes objects such as servers, server groups, customers, and facilities. In the OGFS, these objects are represented as directory structures:

```
/opsw/Customer
/opsw/Facility
/opsw/Group
/opsw/Library
/opsw/Realm
/opsw/Server
. . .
```

The preceding list is not complete. To see the full list, enter `ls /opsw`.

Object attributes

The attributes of an SA object are represented by text files in the `attr` subdirectory. The name of each file matches the name of the attribute. The contents of a file reveals the value of the attribute.

For example, the `/opsw/Server/@/buzz.example.com/attr` directory contains the following files:

```
agentVersion
codeset
createdBy
createdDt
customer
defaultGw
36 Chapter 2
description
discoveredDate
facility
hostName
locale
lockInfo
loopbackIP
managementIP
manufacturer
. . .
```

To display the management IP address of the `buzz.example.com` server, enter the following commands:

```
cd /opsw/Server/@/buzz.example.com/attr
cat managementIP ; echo
```

Custom attributes

Custom attributes are name-value pairs that you can assign to SA objects such as servers. In the OGFS, custom attributes are represented as text files in the `CustAttr` subdirectory. You can create custom attributes in an OGSH session by creating new text files under `CustAttr`. The following example creates a custom attribute named `MyGreeting`, with a value of `hello there`, on the `buzz.example.com` server:

```
cd /opsw/Server/@/buzz.example.com/CustAttr
echo -n "hello there" > MyGreeting
```

For more examples, see the “Managing Custom Attributes” section in the SA 10.50 User Guide.

The self file

The `self` file resides in the directory of an SA object such as a server or customer. This file provides access to various representations of the current object, depending on the format specifier. For details, see "Format specifiers" on page 47.

To list the ID of the `buzz.example.com` server, enter the following commands:

```
cd /opsw/Server/@/buzz.example.com
cat .self:i ; echo
```

For a server, the default format specifier is the name. The following commands display the same output:

```
cat self ; echo
cat .self:n ; echo
```

The next command lists the attributes of a server in the structure format:

```
cat .self:s
```

Primitive values

The following table indicates how primitive values are converted between the API and their string representations in SA CLI methods. Except for Dates, primitive values do not support format specifiers. Dates support ID format specifiers.

Conversion between primitive types and SA CLI Methods

Primitive type	Java equivalent	Output from SA CLI method	Input to SA CLI methods
String	`java.lang.String`	Character string, presented in the encoding of the current session.	Character string, converted to Unicode from the current session encoding.
Number	`byte, short, int, long, float, double`; and their object equivalents	Decimal format, not localized. Scientific notation for very large or small values.	Examples - Decimal: `101, 512.34, -104` Hex: `0x1F32, 0x2e40` Octal: `0543` Scientific: `4.3E4, 6.532e-9, 1.945e+02`
Boolean	`boolean, Boolean`	`true` or `false`	The string “true” and all mixed-case variants evaluate to `true.` All other values evaluate to `false`.
Binary data	`byte[], Byte[]`	Binary string. No conversion from session encoding.	Binary string. No conversion to session encoding.
Date	java.util.Calendar	Date value. By default, presented in this format: `YYYY/MM/DD HH:MM:SS.mmm` The time is presented in UTC. If an ID format specifier is indicated, the value is presented as the number of milliseconds since the epoch, in UTC.	Same as output.

Arrays

The representation of array objects depends on whether they are standalone (an array attribute file or a method return value) or contained in the structure of a complex object.

First, standalone array objects are presented according the underlying type, separated by new-line characters. Within an array element, a new-line character is escaped by \n and a back slash by \\.

Array values can be output or input using any representation supported by the underlying type. For example, by default, the `getDeviceGroups` method lists the groups as names:

```
All Windows Servers
Servers in Austin
Testing Pool
```

If you indicate the ID format specifier, (`.getDeviceGroups:i`) the method displays the IDs of the groups:

```
com.opsware.device.DeviceGroupRef:15960039
com.opsware.device.DeviceGroupRef:10390039
com.opsware.device.DeviceGroupRef:17380039
```

Second, an array contained in the structure of a complex object is represented as a set of name-value pairs, using the attribute as the name. The attribute appears multiple times, once for each element in the array. The order in which the attributes appear determine the order of the elements in the array. The following example shows a structure that contains two attributes, a string called `subject` and a three-element array of numbers called `ranks`:

```
{ subject=”my favorites” ranks=17 ranks=44 ranks=24 }
```

Arrays can also be represented by directories. Within an array directory, each array element has a corresponding file (for primitive types) or subdirectory (for complex types). The name of each entry is the index number of the array element, starting with zero.

For an array that is the attribute of a complex object, you should modify the array by editing its attribute file. This action completely replaces the array with the contents of the edited file.

For an array containing elements that are complex objects, you should modify the array by changing its directory representation. To change an element value, edit the element file. For example, suppose you have an array with five string elements. The `ls` command lists the elements as follows:

```
0 1 2 3 4
```

The following command changes the value of the third element:

```
echo -n "My new value" > 2
```

SA CLI method parameters and return values

This section discusses the details of method context (instance or static), parameter usage, return values, and exit status.

Method context and the self parameter

In the OGFS, a method resides in multiple locations. The location of a method is related to its context, which is either instance or static.

The method with instance context resides in `method` directory of a specific SA object. The method invocation does not require the `self` parameter. The instance of the object affected by the method is implied by the method location. The following example changes the customer of the `d04.example.co`m server:

```
cd /opsw/Server/@/d04.example.com/method
./setCustomer customer:i=9
```

A method with static context resides in a single location under `/opsw/api`. The method invocation requires the self parameter to identify the instance affected by the method. In the following static context example, `self:i` specifies the ID of the managed server:

```
cd /opsw/api/com/opsware/server/ServerService/method
./setCustomer self:i=230054 customer:i=9
```

Passing arguments on the command-line

The command-line arguments are specified as name-value pairs, joined by the equal sign (=). The name-value pairs are separated by one or more white space characters, typically spaces. The names on the command-line match the parameter names of the corresponding Java method in the SA API.

For example, in the SA API, the `setCustomField` method has the following definition:

```
public void setCustomField(CustomFieldReference self,
        java.lang.String fieldName, java.lang.String strValue)...
```

The following SA CLI method example assigns a value to a custom field of the server with ID 3670039:

```
cd /opsw/api/com/opsware/server/ServerService/method
./setCustomField self:i=3670039 \
fieldName="Service Agreement" strValue="Gold"
```

Writer's Note: check the above command by running it (TBD)

As described in the previous section, a method with an instance context does not require the `self` parameter. The following `setCustomField` example is equivalent to the preceding example:

```
cd /opsw/.Server.ID/3670039
./setCustomField \
fieldName="Service Agreement" strValue="Gold"
```

You can specify the command-line arguments in any order. The following two SA CLI method invocations are equivalent:

```
./setCustomField fieldName="My Stuff" strValue="abc"
./setCustomField strValue="abc" fieldName="My Stuff"
```

To specify a null value for a parameter, either omit the parameter or insert a white space after the equal sign. In the following examples, the value of `myParam` is null:

```
./someMethod myField="more info" myParam= anotherParam=9834
./someMethod myField="more info"         anotherParam=9834
```

Specifying the type of a parameter

If a method has an abstract type for a parameter, you must specify the concrete type as well as the value. In the following example, the `com.opsware.folder.FolderRef` type is required:

```
cd /opsw/api/com/opsware/folder/FolderService/method
./remove self:i="com.opsware.folder.FolderRef:730555"
```

If you do not specify the concrete type, the following error message is displayed:

Object type *type-name* is abstract. Specify a concrete sub-type.

Complex objects and arrays as parameters

To pass an argument that is a complex object, enclose the object's attributes in curly braces, as shown in "Structure format specifier syntax " on page 49.

The following example creates a public server group named AllMine. The create method has a single parameter, pattern, which encloses the parent and shortName attributes in curly braces. In this example, getPublicRoot returns 2340555, the ID of the top public group.

```
cd /opsw/api/com/opsware/device/DeviceGroupService/method
./.getPublicRoot:i ; echo
./create “pattern={ parent:i=2340555 shortName=’AllMine’ }”
```

Specify array parameters by repeating the parameter name, once for each array element. For example, the following invocation of the assign method specifies the first two elements in the array parameter named policies:

```
cd /opsw/api/com/opsware/swmgmt
cd SoftwarePolicyService/method
./attachPolicies self:i=4220039 \
policies:i=4400335 policies:i=4400942
```

Overloaded methods

A Java method name is overloaded if multiple methods in the same class have the same name but different parameter lists. With overloaded SA CLI methods, the argument names on the command-line indicate which method to invoke. The setCustomField method, for example, is overloaded to support the setting of different data types. The following two commands invoke different versions of the method:

```
./setCustomField \
fieldName="Service Agreement" strValue="Gold"
./setCustomField \
fieldName=hmp longValue=2245
```

Return values

If the API method underlying an SA CLI method returns a value, then the SA CLI method outputs the value to stdout. As with Unix commands, you can redirect a method’s stdout to a file or assign it to an environment variable.

To change the representation of the return value, insert a leading period and append a format specifier to the method name. The following example returns server references as IDs, instead of the default names:

```
cd /opsw/api/com/opsware/server/ServerService/method
./.findServerRefs:i
```

If you indicate a format specifier that is incompatible with the method's return type, the file system responds with an error.

Exit Status

Like Unix shell commands, SA CLI methods use the exit status (`$?`) to indicate the result of the call. An exit status of zero indicates success; a non-zero indicates an error. SA CLI methods output error messages to `stderr`.

Exit status codes for SA CLI methods

Exit status	Category	Description
0	Success	The method completed successfully.
1	Command-Line Parse Error	The command-line for the method call is malformed and could not be parsed into a set of options (--option[=value]) and parameter values (param=value).
2	Parameter Parse Error	The parameter values could not be parsed into the object types required by the API.
3	API Usage Error	The call failed because of a usage error, such as an invalid parameter value.
4	Access Error	The user does not have permission to perform the operation.
5	Other Error	An error occurred other than those indicated by exit statuses 1- 4.

For example, the following `bash` script checks the exit status of the `getDeviceGroups` method:

```
#!/bin/bash

cd /opsw/Server/@/toro.snv1.corp.example.com/method
./getDeviceGroups
cmnd_exit_status=$?
if [ $cmnd_exit_status -eq 0 ]
then
       echo "The command was successful."
else
       echo "The command failed."
       echo "Exit status = " $cmnd_exit_status
fi
```

An SA CLI method invokes an underlying API method. If the API method throws an exception, the SA CLI method returns a non-zero exit status. When debugging a method call, you might find it helpful to view information about a thrown exception. The `/sys/last-exception` file in the OGFS contains the stack trace of an exception thrown by the most recent API call. After this file has been read, the system discards the file contents.

Search filters and SA CLI methods

Many methods in the SA API accept object references as parameters. To retrieve object references based on search criteria, you invoke methods such as `findServerRefs` and `findJobRefs`. For example, you can invoke `findServerRefs` to search for all servers that have `example.com` in the `hostname` attribute.

Search syntax

Methods such as `findServerRefs` have the following syntax:

```
findobjectRefs filter=’[object-type:]expression’
```

The `filter` parameter includes an expression, which specifies the search criteria. You enclose an expression in either parentheses or curly brackets. A simple expression has the following syntax:

value-object.attribute operator value

This syntax is simplified. For the full definition, see "Filter grammar" on page 216.

Search examples

Most of the SA object types have associated finder methods. This section shows how to use just a few of them. To see how searches are used with other SA CLI methods, see "Sample scripts" on page 65.

Finding servers

Find servers with host names containing `example.com`:

```
cd /opsw/api/com/opsware/server/ServerService/method
./.findServerRefs:i \
filter=’device:{ ServerVO.hostname CONTAINS example.com }’
```

Find servers with a use attribute value of either `UNKNOWN` or `PRODUCTION`:

```
cd /opsw/api/com/opsware/server/ServerService/method
./.findServerRefs:i \
filter=’{ ServerVO.use IN “UNKNOWN” “PRODUCTION” }’
```

The following `bash` script shows how to search for servers, save their IDs in a temporary file, and then specify each ID as the parameter of another method invocation. This script displays the public groups that each Linux server belongs to.

```
#!/bin/bash


TMPFILE=/tmp/server-list.txt

rm -f $TMPFILE


cd /opsw/api/com/opsware/server/ServerService/method


./.findServerRefs:i \

filter='{ ServerVO.osVersion CONTAINS Linux }' > $TMPFILE


for ID in `cat "$TMPFILE"`

do

  echo Server ID: $ID

  ./getDeviceGroups self:i=$ID

  echo

done
```

Finding jobs

The examples in this section return the IDs of jobs such as server audits or policy remediations.

Find the jobs that have completed successfully:

```
cd /opsw/api/com/opsware/job/JobService/method
```

```
./.findJobRefs:i filter='job:{ job_status = "SUCCESS" }'
```

(For a list of allowed values of `job_status`, see “Job Approval Integration” in the SA 10.50 Integration Guide.)

Find the jobs that have completed successfully or with warning:

```
cd /opsw/api/com/opsware/job/JobService/method

./.findJobRefs:i \

filter='job:{ job_status IN "SUCCESS" "WARNING" }'
```

Find the jobs that have been started today:

```
cd /opsw/api/com/opsware/job/JobService/method

./.findJobRefs:i \

filter='job:{ JobInfoVO.startDate IS_TODAY "" }'
```

Find all server audit jobs:

```
cd /opsw/api/com/opsware/job/JobService/method

./findJobRefs \

filter='job:{ JobInfoVO.description = "Server Audit" }'
```

Find the jobs that have run on the server with the ID `280039`:

```
cd /opsw/api/com/opsware/job/JobService/method

./.findJobRefs:i filter='job:{ job_device_id = "280039" }'
```

Find today’s jobs that have failed:

```
cd /opsw/api/com/opsware/job/JobService/method

./.findJobRefs:i \

filter='job:{ (( JobInfoVO.startDate IS_TODAY "" ) \

& ( job_status = "FAILURE" )) }'
```

Finding other objects

This section has examples that search for software policies and packages.

Find the software policies created by the SA user `jdoe`:

```
cd /opsw/api/com/opsware/swmgmt/SoftwarePolicyService/method

./.findSoftwarePolicyRefs:i \
filter=’{ SoftwarePolicyVO.createdBy CONTAINS jdoe }’
```

Find the MSIs with `ismtool` for the Windows 2003 platforms:

```
cd /opsw/api/com/opsware/pkg/UnitService/method

./.findUnitRefs:i \

filter='software_unit:{ ((UnitVO.unitType = "MSI") \

& ( UnitVO.name contains "ismtool" ) \

& ( software_platform_name = "Windows 2003" )) }'
```

Find the Solaris patches named `117170-01`:

```
cd /opsw/api/com/opsware/pkg/solaris/SolPatchService/method

./.findSolPatchRefs:i filter='{name = 117170-01}'
```

Find the folder with the name that includes the string `Test` and with a parent folder named `My Stuff`.

```
cd /opsw/api/com/opsware/folder/FolderService/method

./.findFolders:s \

filter='( ( FolderVO.name CONTAINS "Test" ) \

& (folder_parent_name = "My Stuff" ) )'
```

Searchable attributes and valid operators

Not every attribute of a value object can be specified in a search filter. For example, you can search on `ServerVO.use` but not on `ServerVO.OsFlavor`.

```
To find out which attributes are searchable for a given object type, invoke the
getSearchableAttributes method. The following example lists the attributes of
ServerVO that can be specified in a search expression:
```

```
cd /opsw/api/com/opsware/search/SearchService/method

./getSearchableAttributes searchableType=device
```

The `searchableType` parameter indicates the object type. To determine the allowed values for `searchableType`, enter the following commands:

```
cd /opsw/api/com/opsware/search/SearchService/method

./getSearchableTypes
```

To find out which operators are valid for an attribute, invoke the getSearchableAttributeOperators method. The following example lists valid operators (such as CONTAINS and IN) for the attribute ServerVO.hostname:

```
cd /opsw/api/com/opsware/search/SearchService/method
./getSearchableAttributeOperators searchableType=device \
searchableAttribute=ServerVO.hostname
```

Sample scripts

This section has code listings for simple bash scripts that invoke a variety of SA CLI methods. These scripts demonstrate how to pass method parameters on the command-line, including complex objects and the self parameter. If you decide to copy and paste these example scripts, you will need to change some of the hard-coded object names, such as the d04.example.com server. For tutorial instructions on creating and running scripts within the OGFS, see "SA CLI method tutorial" on page 41.

The script "remediate_policy.sh" on page 70 creates a software policy, adds a package to the policy, and in the last line, installs the package on a managed server by invoking the startFullRemediateNow method.

create_custom_field.sh

This script creates a custom field (virtual column), named TestFieldA attaches the field to all servers, and then sets the value of the field on a single server. Until it is attached, the custom field does not appear in the SA Client. You can create custom fields for servers, device groups, or software policies. To create a custom field, your SA user must belong to a user group with the Manage Virtual Columns permission.

Unlike a custom attribute, a custom field applies to all instances of a type. For an example that creates a custom attribute in the OGFS, see Managing custom attributes in the Server Automation Using Guide on the HPE SSO portal.

The create_custom_field.sh script has the following code:

```
#!/bin/bash
# create_custom_field.sh

cd /opsw/api/com/opsware/custattr/VirtualColumnService/method
```

```
# Create a virtual column.
# Remember the name because you cannot search for the
# displayName.
./create vo=’{ name=TestFieldA type=SHORT_STRING \
displayName="Test Field A" }’

column_id=‘./.findVirtualColumn:i name=TestFieldA‘

echo --- column_id = $column_id

cd /opsw/api/com/opsware/server/ServerService/method

# Attach the column to all servers.
# All servers will have this custom field.
./attachVirtualColumn virtualColumn:i=$column_id

# Get the ID of the server named d04.example.com
devices_id=‘./.findServerRefs:i \
filter=\
’device:{ ServerVO.hostname CONTAINS "d04.example.com" }’‘

echo --- devices_id = $devices_id

# Set the value of the custom field (virtual column) for
# a specific server.
./setCustomField self:i=$devices_id fieldName=TestFieldA \
strValue="This is something."
```

create_device_group.sh

This script creates a static device group and adds a server to the group. Next, the script creates a dynamic group, sets a rule on the group, and refreshes the membership of the group. The last statement of the script lists the devices that belong to the dynamic group.

Here is the script’s code:

```
#!/bin/bash
# create_device_group.sh

cd /opsw/api/com/opsware/device/DeviceGroupService/method

# Get the ID of the public root group (top of hierarchy).
public_root=‘./.getPublicRoot:i‘

# Create a public static group.
./create "vo={ parent:i=$public_root shortName=’Test Group A’ }"

# Get the ID of the group just created.
group_id=‘./.findDeviceGroupRefs:i \
filter=’{ DeviceGroupVO.shortName = "Test Group A" }’ ‘

echo --- group_id = $group_id

cd /opsw/api/com/opsware/server/ServerService/method

# Get the ID of the server named d04.example.com
devices_id=‘./.findServerRefs:i \
filter=\
’device:{ ServerVO.hostname CONTAINS "d04.example.com" }’‘
```

```
echo --- devices_id = $devices_id

cd /opsw/api/com/opsware/device/DeviceGroupService/method

# Add a server to the device group.
./addDevices \
self:i=$group_id devices:i=$devices_id

# Create a dynamic device group.
./create \
"vo={ parent:i=$public_root \
shortName=’Test Dyn B’ dynamic=true }"

# Get the ID of the device group.
dynamic_group_id=‘./.findDeviceGroupRefs:i \
filter=’{ DeviceGroupVO.shortName = "Test Dyn B" }’ ‘

echo --- dynamic_group_id = $dynamic_group_id

# Set the rule so that this group contains servers with
# hostnames containing the string example.com.
# The rule parameter has the same syntax as the filter
# parameter of the find methods.
./setDynamicRule self:i=$dynamic_group_id \
rule=’device:{ ServerVO.hostname CONTAINS example.com }’

# By default, membership in dynamic device groups is refreshed
# once
# an hour, so force the refresh now.
./refreshMembership selves:i=$dynamic_group_id now=true
```

```
# Display the names of the devices that belong to the group.
echo --- Devices in group:
./getDevices selves:i=$dynamic_group_id
```

create_folder.sh

This script creates a folder named `/Test 1`, lists the folders under the root (/) folder, and then creates the subfolder `/Test 1/Test 2`. After creating these folders, you can view them under the Library in the navigation pane of the SA Client.

Here is the code for this script:

```
#!/bin/bash
# create_folder.sh

cd /opsw/api/com/opsware/folder/FolderService/method

# Get the ID of the root (top) folder.
root_id=`./.getRoot:i`

# Create a new folder under the root folder.
./create vo="{ name='Test 1' folder:i=$root_id }"

# Display the names of the folders under the root folder.
./getChildren self:i=$root_id

# Get the ID of the folder "/Test 1"
folder_id=`./.getFolderRef:i path="Test 1"`

# Create a subfolder.
./create vo="{ name='Test 2' folder:i=$folder_id }"

# Get the ID of the folder "/Test 1/Test 2"
```

```
folder_id=`./.getFolderRef:i path="Test 1" path="Test 2"`

echo folder_id = $folder_id
```

remediate_policy.sh

This script creates a software policy named `TestPolicyA` in an existing folder named `Test 2`, adds a package containing `ismtool` to the policy, attaches the policy to a single server (not a group), and then remediates the server. The remediation action launches a job that installs the package onto the server. You can check the progress and results of the job in the SA Client. For examples that search for jobs with SA CLI methods, see "Finding jobs" on page 62.

In this script, in the `create` method of the `SoftwarePolicyService`, the value of the `platforms` parameter is hard-coded. In most of these example scripts, hard-coding is avoided by searching for an object by name. In the case of platforms, searching by the `name` attribute is difficult because if differs from the `displayName` attribute, which is exposed in the SA Client but is not searchable. The easiest way to find a platform ID is by going to the twister and running the `PlatformService.findPlatformRefs` method with no parameters.

The `update` method in this script hard-codes the ID of `softwarePolicyItems`, an object that can be difficult to search for by name if the Software Repository contains many packages with similar names. One way to get the ID is to run the SA Client, search for Software by fields such as File Name and Operating System, open the package located by the search, and note the SA ID in the properties view of the package.

In the following listing, the `update` method has a bad line break. If you copy this code, edit the script so that the `vo` parameter is on a single line.

Here is the source code for the `remediate_policy.sh` script:

```
#!/bin/bash

# remediate_policy.sh


# Get the ID of the folder where the policy will reside.

cd /opsw/api/com/opsware/folder/FolderService/method

folder_id=`./.findFolders:i filter='{ FolderVO.name = "Test 2" }'`


cd /opsw/api/com/opsware/swmgmt/SoftwarePolicyService/method
```

```
# Create a software policy named TestPolicyA.
# This policy resides in the folder located in the preceding findFolders
# call.
# The platform for this policy is Windows 2008 (ID 160076)
./create vo="{ platforms:i=160076 name="TestPolicyA" \
folder:i=$folder_id lifecycle=AVAILABLE }"

policy_id=`./.findSoftwarePolicyRefs:i \
filter='{ SoftwarePolicyVO.name = "TestPolicyA" }'`

echo --- policy_id = $policy_id

# Call the update method to add a package to the software policy.
# The package ID for the "ismtool" msi installer is 4010001.

# Note that "force = true" is required.

./update self:i=$policy_id force=true \
vo='{ softwarePolicyItems:i=com.opsware.pkg.windows.MSIRef:4010001 }'

cd /opsw/api/com/opsware/server/ServerService/method

# Get the ID of the server named d04.opsware.com
devices_id=`./.findServerRefs:i \
filter='device:{ ServerVO.hostname CONTAINS "d04.opsware.com" }'`

echo --- devices_id = $devices_id

# Attach the policy to a single server (not a group).
./attachPolicies self:i=$devices_id \
policies:i=$policy_id
```

```
# Remediate the server to install the package in the policy.
job_id=`./.startFullRemediateNow:i self:i=$devices_id`

echo --- job_id = $job_id
```

remove_custom_field.sh

Although not common in an operational environment, removing custom fields is sometimes necessary in a testing environment. Note that a custom field must be unattached before it can be removed.

Here is the code for `remove_custom_field.sh`:

```
#!/bin/bash
# remove_custom_field.sh

if [ ! -n "$1" ]
   then
   echo "Usage: ‘basename $0‘ <name>"
   echo "Example: ‘basename $0‘ hmp"
   exit
fi

cd /opsw/api/com/opsware/custattr/VirtualColumnService/method

column_id=‘./.findVirtualColumn:i name=$1‘

echo --- column_id = $column_id

cd /opsw/api/com/opsware/server/ServerService/method

# Column must be detached before it can be removed.
./detachVirtualColumn virtualColumn:i=$column_id
```

```
cd /opsw/api/com/opsware/custattr/VirtualColumnService/method

# Remove the virtual column.
./remove self:i=$column_id
```

schedule_audit_task.sh

This script starts an audit task, scheduling it for a future date. With SA CLI methods, date parameters are specified with the following syntax:

```
YYYY/MM/DD HH:MM:SS.sss
```

The method that launches the task, `startAudit`, returns the ID of the job that performs the audit. For examples that search for jobs with SA CLI methods, see "Finding jobs" on page 62.

Here is the code for `schedule_audit_task.sh`:

```
#!/bin/bash
# schedule_audit_task.sh

cd /opsw/api/com/opsware/compliance/sco/AuditTaskService/method

# Get the ID of the audit task to schedule.

audit_task_id=`./.findAuditTask:i \
filter='audit_task:{ (( AuditTaskVO.name BEGINS_WITH "HW check" ) \
& ( AuditTaskVO.createdBy = "gsmith" )) }'`

echo --- audit_task_id = $audit_task_id

# Schedule the audit task for Oct. 16, 2013.
# In the startDate parameter, note that the last delimiter for the time
# is a period, not a colon.
```

```
job_id=`./.startAudit:i self:i=$audit_task_id

schedule:s='{ startDate="2013/10/16 00:00:00.000" }' \

notification:s='{ onFailureOwner="sjones@opsware.com" \

onFailureRecipients="jdoe@opsware.com" \

onSuccessOwner="sjones@opsware.com" \

onSuccessRecipients="jdoe@opsware.com" }'`


echo --- job_id = $job_id
```

Getting usage information on SA CLI methods

In a future release, the SA CLI methods will display usage information. Until then, you can get the necessary information from the API documentation or the OGFS with the techniques described in the following sections.

Listing services

The SA API methods are organized into services. To find out what services are available for SA CLI methods, enter the following commands in an OGSH session:

```
cd /opsw/api/com/opsware

find . -name "*Service"
```

To list the services in the API documentation, specify the following URL in your browser:

`https://`*`occ_host`*`:1032`

The *occ_host* is the IP address or host name of the core server running the Command Center component.

Finding a service in the API documentation

The path of the service in the OGFS maps to the Java package name in the API documentation. For example, in the OGFS, the `ServerService` methods appear in the following directory:

```
/opsw/api/com/opsware/server
```

In the API documentation, the following interface defines these methods:

```
com.opsware.server.ServerService
```

Listing the methods of a service

In the OGFS, you can list the contents of the `method` directory of a service, For example, to display the method names of the `ServerService`, enter the following command:

```
ls /opsw/api/com/opsware/server/ServerService/method
```

In the API documentation, perform the following steps to view the methods of `ServerService`:

In the upper left pane, select `com.opsware.server`.

In the lower left pane, select `ServerService`.

In the main pane, scroll down to view the methods.

Listing th parameters of a method

In the API documentation, perform the steps described in the preceding section. In the Method Detail section of the service interface page, view the parameters and return types. For more information about method parameters, see "Passing arguments on the command-line" on page 57.

Getting information about a value object

The API documentation shows that some service methods pass or return value objects (VOs), which contain data members (attributes). For example, the `ServerService.getServerVO` method returns a `ServerVO` object. To find out what attributes `ServerVO` contains, perform the following steps:

In the API documentation, select the `ServerVO` link. You can find the this link in several places:

- The method signature for `getServerVO`
- The list of classes (lower left pane) for `com.opsware.server`
- On the Index page. A link to the Index page is at the top of the main pane of the API documentation.

- On the ServerVO page, note the getter and setter methods. Each getter-setter pair corresponds to an attribute contained in the value object. For example, getCustomer and setCustomer indicate that ServerVO contains an attribute named customer.

Determining if an attribute can be modified

Only a few object attributes can be modified by client applications. To find out if an attribute can be modified, perform the following steps:

1. In the API documentation, go to the value object page, as described in the preceding section.
2. In the Method Detail section of the setter method, look for “Field can be set by clients.”

For SA objects represented in the OGFS, such as servers and customers, you can determine which attributes are modifiable by checking the access types of the files in the attr directory. The files that have read-write (rw) access types correspond to modifiable attributes. For example, to list the modifiable attributes of a server, enter the following commands:

```
cd /opsw/Server/@/server-name/attr
```

```
ls -l | grep rw
```

Determining if an attribute can be used in a filter query

To find out if an attribute of a value object can be used in a filter query (a search), perform the following steps:

1. In the API documentation, go to the value object page.
2. In the Method Detail section of the getter method that corresponds to the attribute, look for the string, “Field can be used in a filter query.”

From within an OGSH session, to find out if an attribute can be searched on, follow the techniques described in "Searchable attributes and valid operators" on page 64.

Python API access with Pytwist

Overview of Pytwist

Pytwist is a set of Python libraries that provide access to the SA API from managed servers and custom extensions. (The twist is the internal name for the Web Services Data Access Engine.) For managed servers, you can set up Python scripts that call SA APIs through Pytwist so that end users can invoke the scripts as DSEs or ISM controls. Created by HPE SA Professional Services, custom extensions are Python scripts that run in the Command Engine (way). Pytwist enables custom extensions to access recent additions to the SA data model, such as folders and software policies, which are not accessible from Command Engine scripts.

This topic is intended for developers and consultants who are already familiar with the SA data model, custom extensions, Agents, and the Python programming language.

Setup for Pytwist

Before trying out the examples in this section, make sure that your environment meets the following setup requirements, as detailed in the following sections.

Supported platforms for Pytwist

Pytwist is supported on managed servers and core servers. For a list of operating systems supported for these servers.

Pytwist relies on Python version 2.7.10, the version used by SA Agents and custom extensions.

Unlike Web Services and Java RMI clients, a Pytwist client relies on internal SA libraries. If your client program needs to access the SA API from a server that is not a managed or core server, then use a Web Services or Java RMI client, not Pytwist.

Access requirements for Pytwist

Pytwist needs to access port 1032 of the core server running the Web Services Data Access Engine. By default, the engine listens on port 1032.

Installing Pytwist libraries

The pytwist libraries need not be installed as they are part of the agent libraries.

Pytwist examples

The Python code examples in this section show how to get information from managed servers, create folders, and remediate software policies. Each Pytwist example performs the following operations:

1. Import the packages.

   When importing objects of the SA API name space, such as `Filter`, the path includes the Java package name, preceded by `pytwist`. Here are the `import` statements for the `get_server_info.py` example:

   ```
   import sys
   from pytwist import *
   from pytwist.com.opsware.search import Filter
   ```

2. Create the `TwistServer` object:

   ```
   ts = twistserver.TwistServer()
   ```

   See TwistServer method syntax for information about the method's arguments.

3. Get a reference to the service.

   The Python package name of the service is the same as the Java package name, but without the leading `opsware.com`. For example, the Java `com.opsware.server.ServerService` package maps to the Pytwist `server.ServerService`:

   serverservice = ts.server.ServerService

4. Invoke the SA API methods of the service:

   ```
   filter = Filter()
   . . .
   servers = serverservice.findServerRefs(filter)
   . . .
   for server in servers: vo = serverservice.getServerVO(server)
   . . .
   ```

get_server_info.py

This script searches for all managed servers with host names containing the command-line argument. The search method, `findServerRefs`, returns an array of references to server persistent objects. For each reference, the `getServerVO` method returns the value object (VO), which is the data representation that holds the server’s attributes. Here is the code for the `get_server_info.py` script:

```
#!/opt/opsware/agent/bin/python
# get_server_info.py

# Search for servers by partial hostname.

import sys
from pytwist import *
from pytwist.com.opsware.search import Filter

# Check for the command-line argument.
if len(sys.argv) < 2:
       print "You must specify part of the hostname as the search target."
       print "Example: " + sys.argv[0] + " " + "opsware.com"
       sys.exit(2)
# Construct a search filter.
filter = Filter()
filter.expression = 'device_hostname *=* "%s" ' % (sys.argv[1])

# Create a TwistServer object.
ts = twistserver.TwistServer()

# Get a reference to ServerService.
serverservice = ts.server.ServerService

# Perform the search, returning a tuple of references.
servers = serverservice.findServerRefs(filter)

if len(servers) < 1:
       print "No matching servers found"
       sys.exit(3)

# For each server found, get the server’s value object (VO)
# and print some of the VO’s attributes.
for server in servers:
       vo = serverservice.getServerVO(server)
       print "Name: " + vo.name
       print "Management IP: " + vo.managementIP
       print "OS Version: " + vo.osVersion
```

create_folder.py

This script creates a folder named `/TestA/TestB` by invoking the `createPath` method. Note that the `path` parameter of `createPath` does not contain slashes. Each string element in `path` indicates a level in the folder. Next, the script retrieves and prints the names of all folders directly below the root folder. The listing for the `create_folder.py` script follows:

```
#!/opt/opsware/agent/bin/python
# create_folder.py

# Create a folder in SA.

import sys
from pytwist import *

# Create a TwistServer object.
ts = twistserver.TwistServer()

# Get a reference to FolderService.
folderservice = ts.folder.FolderService

# Get a reference to the root folder.
rootfolder = folderservice.getRoot()
# Construct the path of the new folder.
path = 'TestA', 'TestB'

# Create the folder /TestA/TestB relative to the root.
folderservice.createPath(rootfolder, path)

# Get the child folders of the root folder.
rootchildren = folderservice.getChildren(rootfolder,
'com.opsware.folder.FolderRef')

# Print the names of the child folders.
for child in rootchildren:
       vo = folderservice.getFolderVO(child)
       print vo.name
```

remediate_policy.py

This script creates a software policy, attaches it to a server, and then remediates the policy. Several names are hard-coded in the script: the platform, server, and parent folder. Optionally, you can specify the policy name on the command-line, which is convenient if you run the script multiple times. The platform of the software policy must match the OS of the packages contained in the policy. Therefore, if you change the hard-coded platform name, then you also change the name in `unitfilter.expression`.

The following listing has several bad line breaks. If you copy this code, be sure to fix the bad line breaks before running it. The comment lines beginning with "NOTE" point out the bad line breaks.

```
#!/opt/opsware/agent/bin/python
# remediate_policy.py

# Create, attach, and remediate a software policy.

import sys
from pytwist import *
from pytwist.com.opsware.search import Filter
from pytwist.com.opsware.swmgmt import SoftwarePolicyVO

# Initialize the names used by this script.
foldername = 'TestB'
platformname = 'Windows 2003'
servername = 'd04.example.com'
# If a command-line argument is specified,
# use it as the policy name
if len(sys.argv) == 2:
       policyname = sys.argv[1]
else:
       policyname = 'TestPolicyA'

# Create a TwistServer object.
ts = twistserver.TwistServer()
ts.authenticate("SAUser", "SAPassword")

# Get the references to the services used by this script.
folderservice = ts.folder.FolderService
swpolicyservice = ts.swmgmt.SoftwarePolicyService
serverservice = ts.server.ServerService
unitservice = ts.pkg.UnitService
platformservice = ts.device.PlatformService

# Search for the folder that will contain the policy.
folderfilter = Filter()
folderfilter.expression = 'FolderVO.name = %s' % foldername
folderrefs = folderservice.findFolderRefs(folderfilter)
if len(folderrefs) == 1:
       parent = folderrefs[0]
elif len(folderrefs) < 1:
       print "No matching folders found."
       sys.exit(2)
```

```
else:
       print "Non-unique folder name: " + foldername
       sys.exit(3)
# Search for the reference to the platform "Windows Server 2003."
platformfilter = Filter()
platformfilter.objectType = 'platform'
# Because the platform name contains spaces,
# it’s enclosed in double quotes
# NOTE: The following code line has a bad line break.
# The assignment statement should be on a single line.
platformfilter.expression = 'platform_name = "%s"' % platformname
platformrefs = platformservice.findPlatformRefs(platformfilter)

if len(platformrefs) == 0:
       print "No matching platforms found."
       sys.exit(4)

# Search for the references to some software packages.
unitfilter = Filter()
unitfilter.objectType = 'software_unit'
# NOTE: The following code line has a bad line break.
# The assignment statement should be on a single line.
unitfilter.expression = '((UnitVO.unitType = "MSI") & ( UnitVO.name contains
"ismtool" ) & ( software_platform_name = "Windows 2003" ))'
unitrefs = unitservice.findUnitRefs(unitfilter)

# Create a value object for the new software policy.
vo = SoftwarePolicyVO()
vo.name = policyname
vo.folder = parent
vo.platforms = platformrefs
vo.softwarePolicyItems = unitrefs

# Create the software policy.
swpolicyvo = swpolicyservice.create(vo)

# Search by hostname for the reference to a managed server.
serverfilter = Filter()
serverfilter.objectType = 'server'
# NOTE: The following code line has a bad line break.
# The assignment statement should be on a single line.
serverfilter.expression = 'ServerVO.hostname = %s' % servername
serverrefs = serverservice.findServerRefs(serverfilter)

if len(serverrefs) == 0:
       print "No matching servers found."
       sys.exit(5)

# Create an array that has a reference to the
# newly created policy.
swpolicyrefs = [1]
swpolicyrefs[0] = swpolicyvo.ref

# Attach the software policy to the server.
swpolicyservice.attachToPolicies(swpolicyrefs, serverrefs)

# Remediate the policy and the server.
# NOTE: The following code line has a bad line break.
# The assignment statement should be on a single line.
jobref = swpolicyservice.startRemediateNow(swpolicyrefs, serverrefs)
print "The remediation job ID is %d" % jobref.id
```

Virtualization Pytwist examples

This section provides examples of creating and deploying virtual machines (VMs) using SA API. For more examples about Virtualization, see the Server Automation Using Guide on the HPE SSO portal.

createVM_WithOSBP.py

This basic example creates a VM on a VMware vCenter using CD boot with static IP configuration.

All properties have not been set in these examples. Please refer to API documentation (javadocs) to understand and set the properties for your use case.

```
#!/opt/opsware/agent/bin/python
from pytwist import twistserver
from pytwist.com.opsware.locality import CustomerRef, RealmRef
from pytwist.com.opsware.osprov import OSBuildPlanRef
from pytwist.com.opsware.pkg import UnknownPkgRef
from pytwist.com.opsware.v12n import AdapterIPSettings, V12nHypervisorRef, \
       V12nHypervisorService, V12nInventoryFolderRef, V12nResourcePoolRef, \
       V12nResourcePoolRef, V12nVIManagerService, VirtualCpuConfig,
VirtualDevice, \
       VirtualDeviceChangeConfig, VirtualDeviceTypeConstant,
VirtualHardwareConfigSpec, \
       VirtualMemoryConfig, VirtualServerCDProvisioningSpec,
VirtualServerComputeSpec, \
       VirtualServerConfigSpec, VirtualServerCreateSpec,
VirtualStorageDeviceConstant, \
       VirtualStorageDeviceHWConfig
from pytwist.com.opsware.v12n.vmware import V12nDatastoreRef, \
       VmwareVirtualInterfaceBacking, VmwareVirtualNicHWConfig, \
       VmwareVirtualServerDetails, VmwareVirtualServerStorageSpec, \
       VmwareVirtualStorageFileBacking
import time


# This is a bare bones example of creating a Virtual Machine on a VMware
# vCenter while booting from CD with Static IP configuration. It also
# provisions the Virtual Machine with the give OS Build Plan. For more
# detailed information please refer to the java doc. All the properties have
# not been set in the example below, please review the java doc to understand
# and set the properties for your use case.
# This method constructs the create specification to create the Virtual
# Machine and provision it.
def constructCreateSpec():

       # Construct VmwareVirtualServerDetails
       detail = VmwareVirtualServerDetails()
       # Virtual Machine Name
       detail.name = "Test VM"
       # Description for the Virtual Machine
       detail.description = "Sample test create VM"
       # This is the key for the guest operating system that will installed on
       # the Virtual Machine.
       # V12nVIManagerService.getGuestOSList() provides the supported list for
       # the given V12n Manager and hypervisor.
       detail.guestId = "rhel6Guest"
       # This is folder where the VM will reside in you can see the list of
       # folders at V12nInventoryFolderService.findV12nInventoryFolderRefs() it
       # is the inventory location of the Virtual Machine
       folder = V12nInventoryFolderRef(2020001)
       detail.inventoryFolderRef = folder


       # Configure the number of Virtual processors on the Virtual Machine
       cpuConfig = VirtualCpuConfig()
       cpuConfig.virtualCpuCount = 1

       # Configure the Memory for the Virtual Machine
       memoryConfig = VirtualMemoryConfig()
       memoryConfig.size = 1024*1024*1024

       # Configure NICs
       # Construct the virtual device of type network i.e a NIC
       virtualNetworkDevice = VirtualDevice()
       virtualNetworkDevice.type = VirtualDeviceTypeConstant.NETWORK
       # A unique identifier for the virtual device
       virtualNetworkDevice.key = "4001"
       backingNetwork = VmwareVirtualInterfaceBacking()
       # This is the port group that the nic will be assigned to
       backingNetwork.portGroup = "VLAN 625"

       hwConfigNetwork = VmwareVirtualNicHWConfig()
       # The kind of network adapter to use, other options are listed in
       # VmwareVirtualNicHWConfig
       hwConfigNetwork.adapterType = VmwareVirtualNicHWConfig.E1000
       hwConfigNetwork.macAddressIsDynamic = True

       virtualNetworkDevice.hwConfig = hwConfigNetwork
       virtualNetworkDevice.backingInfo = backingNetwork
       virtualNetworkDevice.connected = True
       virtualNetworkDevice.startConnected = True

       # Configure Hard Disk
       virtualDiskDevice = VirtualDevice()
       virtualDiskDevice.type = VirtualDeviceTypeConstant.STORAGE

       backingStorage = VmwareVirtualStorageFileBacking()

       # This is Ref for the data store on the hypervisor where the VM will be
       # hosted. The list of datastores associated with the Hypervisors are
       # listed at V12nHypervisorService.getV12nHypervisorVO() under storage
       # config
       dataStoreRef = V12nDatastoreRef(90001)
       backingStorage.datastore = dataStoreRef
       backingStorage.lazyAllocation = True

       hwConfigStorage = VirtualStorageDeviceHWConfig()
       hwConfigStorage.capacity = 10*1024*1024*1024
       hwConfigStorage.usageType =
VirtualStorageDeviceConstant.USAGE_TYPE_DISK_DRIVE

       virtualDiskDevice.hwConfig = hwConfigStorage
       virtualDiskDevice.backingInfo = backingStorage

       # Add both the virtual devices to be created, i.e. the hard disk and the
       # nic
       virtualDvcs_toAdd = []
       virtualDvcs_toAdd.append(virtualNetworkDevice)
       virtualDvcs_toAdd.append(virtualDiskDevice)
       deviceChange = VirtualDeviceChangeConfig()
       deviceChange.addList = virtualDvcs_toAdd

       # Finalize the Config Spec
       configSpec = VirtualServerConfigSpec()
       configSpec.detail = detail
       configSpec.virtualHardware = VirtualHardwareConfigSpec()
       configSpec.virtualHardware.cpuConfig = cpuConfig
       configSpec.virtualHardware.memoryConfig = memoryConfig
       configSpec.virtualHardware.deviceChange = deviceChange

       # Constructing the Compute Spec
       computeSpec = VirtualServerComputeSpec()
       # This is the hypervisor hosting the VM
       hypervisorRef = V12nHypervisorRef(2030001)
       computeSpec.computeProviderRef = hypervisorRef
       # This is resource pool on the hypervisor/cluster that the VM belongs to
       # It can be retrieved by using hypervisorVO.children or the Cluster
       # children
       resourcePool = V12nResourcePoolRef(2040001)
```

```
computeSpec.resourcePoolRef = resourcePool

storageSpec = VmwareVirtualServerStorageSpec()
storageSpec.datastore = dataStoreRef

# This example deals with provisioning a VM through CD boot and with
# static IP configuration. The example deals setting the boot ISO and
# network information to be used.
# All the information for this is contained in the
# VirtualServerCDProvisioningSpec

# Set all the network information
gateways =[]
gw ="192.168.135.33"
gateways.append(gw)
dnsServers =[]
dnsServer = "192.168.2.13"
dnsServers.append(dnsServer)


interfaces =[]
interface = AdapterIPSettings()

# Construct the network interface
interface.useDHCP=False
# Note this is the virtual device we have created above, we use the same
# device key to indicate to provisioning which virtual device is to be
# used for provisioning
interface.virtualDeviceKey="4001"
interface.gateways=gateways
interface.ipAddress="192.168.135.45"
interface.netmask="255.255.255.224"
interface.dnsServerList=dnsServers

interfaces.append(interface)

# This is the boot ISO Ref that will be used to get the server into
# maintenance mode
# The name and the id need to match the packages on the core.
# Use the UnitService.findUnitRefs() to find the boot ISO's
bootISORef = UnknownPkgRef(5340001)
bootISORef.name="HPSA_linux_boot_cd.iso"
# The realm assigned to the Virtual Machine will be the realm of the
# Virtualization Service
realmRef = RealmRef(30001)
# The OS Build Plan that needs be run on the Virtual Machine after the VM
# has been created.
osbpRef = OSBuildPlanRef(580001)
```

```
        provisioningSpec = VirtualServerCDProvisioningSpec()

        provisioningSpec.bootISORef = bootISORef
        provisioningSpec.interfaces = interfaces
        provisioningSpec.realmRef = realmRef
        provisioningSpec.oSBuildPlanRef = osbpRef

        # Finally put together all the information to be set on the Create
        # Specification
        createSpec = VirtualServerCreateSpec()
        createSpec.configSpec = configSpec
        createSpec.computeSpec = computeSpec
        createSpec.storageSpec = storageSpec

        createSpec.provisioningSpec = provisioningSpec
        #Set the customer to be associated with the Virtual Machine
        customer = CustomerRef(9)
        createSpec.setCustomerRef(customer)
        return createSpec

def createVirtualMachine():
        twist = twistserver.TwistServer()
        twist.authenticate("hp", "opsware")
        vmService = twist.v12n.V12nVirtualServerService
        createSpec = constructCreateSpec()
        jobRef = vmService.startCreate(createSpec,4*60*60,"Sample create
VM",None, None)


createVirtualMachine()
```

deployVM.py

This basic example shows how to deploy a VM from a VM template on VMware vCenter and customize the guest OS of the deployed VM.

All properties have not been set in these examples. Please refer to API documentation (javadocs) to understand and set the properties for your use case.

```
#!/opt/opsware/agent/bin/python
from pytwist import twistserver
from pytwist.com.opsware.locality import CustomerRef, RealmRef
from pytwist.com.opsware.osprov import OSBuildPlanRef
from pytwist.com.opsware.pkg import UnknownPkgRef
from pytwist.com.opsware.v12n import AdapterIPSettings, V12nHypervisorRef, \
        V12nHypervisorService, V12nInventoryFolderRef, V12nResourcePoolRef, \
       V12nResourcePoolRef, V12nVIManagerService, VirtualCpuConfig,
VirtualDevice, \
       VirtualDeviceChangeConfig, VirtualDeviceTypeConstant,
VirtualHardwareConfigSpec, \
       VirtualMemoryConfig, VirtualServerCDProvisioningSpec,
VirtualServerComputeSpec, \
       VirtualServerConfigSpec, VirtualServerCreateSpec,
VirtualStorageDeviceConstant, \
       VirtualStorageDeviceHWConfig, V12nVirtualServerTemplateRef, \
       VirtualServerCloneSpec, VirtualServerGuestCustomizationSpec
from pytwist.com.opsware.v12n.vmware import V12nDatastoreRef, \
       VmwareVirtualInterfaceBacking, VmwareVirtualNicHWConfig, \
       VmwareVirtualServerDetails, VmwareVirtualServerStorageSpec, \
       VmwareVirtualStorageFileBacking
import time

# This is a bare bones example of deploying a Template VMware vCenter. It
# deploys the template and then guest customizes the deployed Virtual Machine.
# For more detailed information please refer to the java doc. All the
# properties have not been set in the example below, please review the java
# doc to understand and set the properties for your use case.

# This method constructs the deploy specification to deploy the Template and
# customizes it.
def constructDeploySpec(sourceTemplateVO):

       # Construct the Deploy Spec
       clonespec = VirtualServerCloneSpec()

       clonespec.computeSpec = VirtualServerComputeSpec()
       # This is the hypervisor hosting the VM
       targetHypervisorRef = V12nHypervisorRef(2030001)
       clonespec.computeSpec.computeProviderRef = targetHypervisorRef

       computeSpec = VirtualServerComputeSpec()
       # This is the resource pool on the hypervisor/cluster that the VM belongs to
       # It can be retrieved by using hypervisorVO.children or the Cluster
       # children
       targetResourcePoolRef = V12nResourcePoolRef(2040001)
       computeSpec.resourcePoolRef = targetResourcePoolRef
       clonespec.computeSpec.resourcePoolRef = targetResourcePoolRef

       storageSpec = VmwareVirtualServerStorageSpec()
       dataStoreRef = V12nDatastoreRef(90001)
       storageSpec.datastore = dataStoreRef
       clonespec.storageSpec = storageSpec
       # Construct VmwareVirtualServerDetails
       detail = VmwareVirtualServerDetails()
       # Virtual Machine Name
       detail.name = "Test Deploy VM"
       # Description for the Virtual Machine
       detail.description = "Sample Deploy create VM"

       # This is the folder where the VM will reside in. You can see the list of
       # folders at V12nInventoryFolderService.findV12nInventoryFolderRefs(). It
       # is the inventory location of the Virtual Machine
       targetFolderRef = V12nInventoryFolderRef(2020001)
       detail.inventoryFolderRef = targetFolderRef
       configSpec = VirtualServerConfigSpec()
       configSpec.detail = detail
       clonespec.configSpec=configSpec

       # Create the Guest Customization Spec, this is needed to customized the
       # deployed VM so that it does not use the network settings and host name
       # of the source template
       # In this example all the interfaces are set to DHCP but you can
       # customize each of the interfaces by either providing static or DHCP
       # configuration details
       interfaces = createInterfaces(sourceTemplateVO)
       # The realm assigned to the Virtual Machine will be the realm of the
       # Virtualization Service
       realmRef = RealmRef(30001)
       clonespec.guestCustomizationSpec =
createGuestCustomizationSpec("testDeployVM",realmRef,interfaces)
       clonespec.setPowerOn(True)
       # Set the customer to be associated with the Virtual Machine
       customerRef = CustomerRef(9)
       clonespec.customerRef = customerRef
       return clonespec

def createGuestCustomizationSpec(newVmNameVal,realmRef,interfaces):
       gcSpec = VirtualServerGuestCustomizationSpec()
       gcSpec.computerName = newVmNameVal
       gcSpec.interfaces = interfaces
       gcSpec.realmRef = realmRef
       return gcSpec

def createInterfaces(virtualServerVO):
       interfaces = []
       virtualDevices = virtualServerVO.virtualHardware.deviceList
       vNICs = [vd for vd in virtualDevices if vd.type ==
VirtualDeviceTypeConstant.NETWORK]
       for vNIC in vNICs:
               intf = AdapterIPSettings()
               intf.useDHCP = True
               intf.hardwareAddress = vNIC.hwConfig.macAddress
               intf.virtualDeviceKey = vNIC.key
               interfaces.append(intf)
```

```
        return interfaces


def deployVirtualMachine():
        twist = twistserver.TwistServer()
        twist.authenticate("hp", "opsware")
        vmTemplateService = twist.v12n.V12nVirtualServerTemplateService
        vmService = twist.v12n.V12nVirtualServerBaseService
        sourceTemplateRef = V12nVirtualServerTemplateRef(1520001)
        sourceTemplateVO =
vmService.getV12nVirtualServerBaseVO(sourceTemplateRef)
        deploySpec = constructDeploySpec(sourceTemplateVO)
        jobRef =
vmTemplateService.startDeploy(sourceTemplateRef,deploySpec,30*60,"Sample
Deploy VM",None, None);

deployVirtualMachine()
```

Pytwist details

This section describes the behavior and syntax that is specific to Pytwist.

Authentication modes

The authentication mode of a Pytwist client is important because it affects the SA features and the resources that the client can access. A Pytwist client can run in one of the following modes:

- **Authenticated**: The client has called the `authenticate(username, password)` method on a `TwistServer` object. After calling the `authenticate` method, the client is authorized as the SA user specified by the `username` parameter, much like an end user who logs onto the SA Client.
- **Not Authenticated**: The client has not called the `TwistServer.authenticate` method. On a managed server, the client is authenticated as if it is the device that controls the Agent certificate. When used within a custom extension, a non-authenticated Pytwist client needs access to the Command Engine certificate. For more information on custom extensions and certificates, contact your technical support representative.

TwistServer method syntax

The `TwistServer` method configures the connection from the client to the Web Services Data Access Engine. (For sample invocations, see Pytwist examples.) All of the arguments of `TwistServer` are optional. The following table lists the default values for the arguments.

Arguments of the TwistServer method

Argument	Description	Default
host	The hostname to connect to.	twist
port	The port number to connect to.	1032
secure	Whether to use https for the connection. Allowed values: 1 (true) or 0 (false).	1
ctx	The SSL context for the connection.	None. (See also "Authentication modes" above.)

When the TwistServer object is created, the client does not establish a connection with the server. Therefore, if a connectivity problem occurs, it is not encountered until the client calls authenticate or an SA API method.

Error handling

If the TwistServer.authenticate method or an SA API method encounters a problem, a Python exception is raised. You can catch these exceptions in an except clause, as in the following example:

```
# Create the TwistServer object.
ts = twistserver.TwistServer(’localhost’)
# Authenticate by passing an SA user name and password.
try:
       ts.authenticate(’jdoe’, ’secretpass’)
except:
       print "Authentication failed."
       sys.exit(2)
```

Mapping Java package names and data types to Pytwist

The Pytwist interface is for Python, but the SA API is written in Java. Because of the differences between two programming languages a Pytwist client must follow the mapping rules described in this section.

In the SA API documentation, Java package names begin with com.opsware. When specifying the package name in Pytwist, insert pytwist at the beginning, for example:

```
from pytwist.com.opsware.compliance.sco import *
```

The SA API documentation specifies method parameters and return values as Java data types. The following table shows how to map the Java data types to Python for the API method invocations in Pytwist.

Mapping data types from Java to Python

Java data type in SA API	Python data type in Pytwist
Boolean	An integer 1 for true or the integer 0 for false.
Object[]	As input parameters to API method calls, object arrays can be either Python tuples or

Mapping data types from Java to Python, continued

Java data type in SA API	Python data type in Pytwist
(object array)	lists. As output from API method calls, object arrays are returned as Python tuples.
Map	Dictionary
`Date`	A `long` data type representing the number of milliseconds since epoch (midnight on January 1, 1970).

Automation Platform Extensions (APX)

This topic describes how to create and manage Automation Platform Extensions (APX), commonly just called *extensions*. APXs provide a framework that allows anyone familiar with script-based programming tools such as shell scripts, Python, Perl, and PHP, to extend the functionality of SA and create applications that are tightly integrated into SA. SA provides two types of APXs:

- **Program APXs** (also called **Script APXs**) run in the Global File System (OGFS) and can use all of the OGFS functionality. You can use typical programming practices to leverage the SA API and access a core's Managed Servers to implement new custom functionality. For example, you could write an APX that gathers BIOS information from managed servers and populates custom fields using shell commands. See "Program APXs" on page 98.
- **Web APXs** allow you to create a web-based application, where either an Apache 2.x process or a CGI/PHP script is called using GET or POST URL. Web APXs can contain static web resources such as images, and can employ CGI or PHP for dynamic content generation. See "Web APXs" on page 99.

APXs allow you to access data about your managed environment and share and process that data with web applications, scripts, programs and other applications. Below are some of the benefits of APXs:

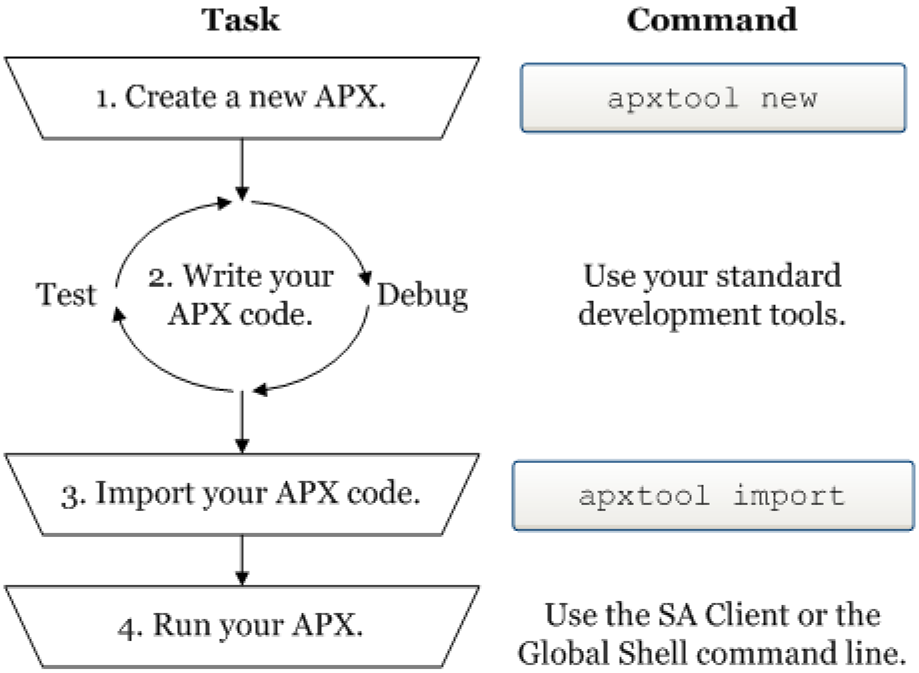
- Listed in the SA Library and can be used from the SA Client.
- Uniquely identified and managed through versioning.
- Secure because they take full advantage of SA's security model. When needed, APXs can securely and temporarily escalate a user's permissions beyond the normal defaults during the APX session.
- Scalable within and across SA cores.
- You can schedule them to be pushed automatically to servers.
- Auditable.
- Able to persist through an upgrade of the SA platform. APXs do not have to be rewritten after an upgrade.

For information on using APX extensions, see the Running Extensions to SA section in the SA 10.50 User Guide. See also the SA Global Shell section in the SA 10.50 User Guide because you can also run APX extensions from the SA Global Shell.

Creating an APX

The following diagram shows the basic steps to creating an APX and the corresponding commands to use. For a tutorial on how to create a web APX, see "Tutorial: Creating a Web Application APX" on page 121. For a tutorial on how to create a program APX, see "Tutorial: Creating a program APX" on page 129.

Creating an APX



1. To create a new APX, use the `apxtool new` command. This command creates a set of template files you can edit to create your own APX.

   You can optionally register your new APX with the `apxtool new` command. Registering your APX reserves the name of your APX in SA. If you do not register your APX at this step, you can register it with the `apxtool import` command in step 3 below.

   See "apxtool command" on page 107.

2. After creating APX template files, develop your APX code by modifying the template files created by the `apxtool new` command and possibly adding your own files. You can test your APX code to make sure it is running correctly.

3. When your APX code is tested, you must import it into SA with the `apxtool import` command.
4. Run your APX either from the SA Client or from the Global Shell command line.

   - From the SA Client: Select **Library** > **By Type tab** > **Extensions** > **Program**. Select an APX. Select the **Actions** > **Run menu**.
   - From the Global Shell command line: Open the Global Shell from the SA Client by selecting the Tools > Global Shell menu. Run your APX by entering the command `/opsw/apx/bin/<APX name>`.

     For more information, see Running Extensions to SA and the SA Global Shell sections in the Server Automation Using Guide on the HPE SSO portal.

To create an APX extension that is intended to run on VMware ESXi servers, the APX extension must communicate with the ESXi server remotely using its web services interface. For more information on VMware ESXi servers, see the Virtual Server Management section in the Server Automation Using Guide on the HPE SSO portal.

Program APXs

Program APXs, also called Script APXs, are similar to shell commands and are implemented as OGFS server scripts. You can invoke them from the OGFS command line and pass input arguments to them using STDIN or command-line arguments. Their output goes to STDOUT and STDERR.

Program APXs are executed inside a Global Shell (OGSH) session and have access to all OGSH features permissible to the user who invokes the APX. This includes rosh, CLI, OGFS, and more. You can write Program APXs using any script-based tool, such as shell script, Python, Perl, and so on.

You can invoke Program APXs from the OGSH command prompt. Typically, Program APXs are executed synchronously, meaning the shell prompt does not return until the Program APX returns. APXs cannot be scheduled as recurring jobs in either the twister or in OGFS.

Program APXs are located in the OGFS directory `/opsw/apx/bin`.

During an interactive OGSH session, a user only sees those Program APXs in /opsw/apx/bin that they have permission to execute. Attempting to invoke a Program APX for which a user has no execution permission results in a `File Not Found` error from the shell.

A Program APX can also be invoked by other Web APXs or Program APXs. For example, a CGI program or PHP script from a Web APX can invoke a Program APX.

Web APXs

Web APXs are implemented using CGI programs or PHP scripts. These CGI programs and PHP scripts are executed inside a user-specific OGSH session. They may access SA facilities such as rosh, the SA API, CLI, or any commands allowable from within an OGSH session. Web APXs are served by a built-in Apache web server with a PHP module enabled.

You can access Web APXs in two ways: using a stand-alone web browser such as Internet Explorer or Firefox, or from the SA Client. Microsoft ActiveX is not supported.

Invoking a Web APX from a stand-alone Web browser the first time will trigger a login dialog that requires verification of the SA user credentials. Invoking a Web APX from the SA Client does not require additional login. Web APXs can be used to build user Interfaces for custom customer applications.

To launch APXs using Microsoft Internet Explorer versions 6 and 7 on Windows Server 2003, 2008 and 2012 with Enhanced Security Configuration enabled, the SA Client URL must first be added to Internet Explorer’s trusted site list.

APX user roles

There are three general roles of APX users as shown in the following table:

APX user roles

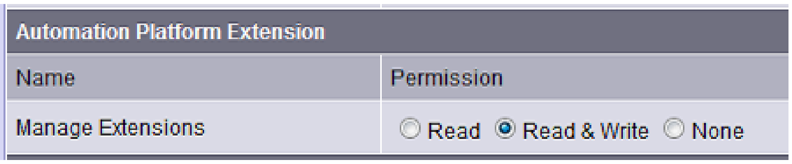
User role	Description
End User	Runs APXs. This user typically does not have permission to modify an APX or see its content.
APX Developer	Creates and publishes APXs. This class of users can import and export APXs, and can modify APX content.
APX Administrator	Determines APXs users are permitted to run. These users assign executable permission to run an APX by managing folder permissions. APX Administrators may not have permission to modify the APX itself, but can have the permission to view APX content in order to determine which APXs to make executable.

APX permissions

APXs requires that you have the SA Client Feature permission **Manage Extensions**. A user group can be given one of the permissions:

- Manage Extensions: Read
- Manage Extensions: Read & Write
- Manage Extensions: None

APX feature permissions



Automation Platform Extension	
Name	Permission
Manage Extensions	○ Read ● Read & Write ○ None

These feature permissions apply only to APX developers and administrators, they do not apply to those users who only need to run APXs.

- **Read** permission grants the ability to display the APX source contents or to export (download) the APX source archives.
- **Read & Write** permission grants the ability to modify the contents of an APX in addition to read access.
- **None** permission denies all access to the APX source.

In addition to the SA Client Feature **Manage Extensions** permission, folder permissions (list, read, write, execute) must be used to determine which APXs a user has access to.

APX permissions

Permission	Description
List	Permission to list the system’s APXs.
Read	Permission to view APX contents.
Write	Permission to modify APX content and to import and export APXs.
Execute	Permission to run APXs and view APX properties.

The following table shows a matrix of how permissions are determined based on the combination of the Manage Extensions feature permissions and folder permissions.

APX permission matrix

	Manage Extensions Permission:		
Folder Permission:	**Read**	**Read & Write**	**None**
List	List APXs	List APXs	List APXs
Read	Export APXs	Export APXs	List APXs
Write	Export APXs	Import, export APXs	List APXs
Execute	Run APXs	Run APXs	Run APXs

Like other SA features, you can grant a user access to an APX and specify to which managed servers and/or policies the user can apply the APX.

If a user attempts to access a Web APX for which he does not have execution permission, the Web browser will receive an HTTP 403 Forbidden return code.

For more information on SA permissions, see the Server Automation Administration Guide on the HPE SSO portal.

Permission escalation

When executing an APX, the user has only the privileges to access resources and operations granted in SA. However, in some cases, it will be necessary to temporarily grant the user *escalated permissions*, privileges beyond the SA privileges, while executing an APX. You can explicitly grant certain privileges to users, over-and-above their default SA privileges, on a temporary basis while running an APX. Permission escalation is transparent to the user running the APX.

For example, you may want a user to be able to run a BIOS information gathering application on a managed server, but the user does not have the permissions granted to do so. You can write an APX for a user without the privileges required to run the BIOS gathering application that temporarily grants that user the required privileges. The user’s privileges return to the default after the APX ends its run.

Privilege escalation is specified in the file apx.perm file. For more information, see "The APX permissions escalation configuration file - apx.perm" on page 118.

APX structure

An APX has the following attributes:

- `APX type`: Either Program APX (also called Script APX) or Web APX.
- `APX unique name`: This is the full name of the APX that must be unique. For example, com.hpe.sa.RestartMyApp.
- `APX display name`: This is usually a shorter name than the APX unique name. For example, RestartMyApp.
- `APX version`: You can maintain multiple versions of your APX by setting a version string or you can let SA manage versions for you automatically. The APX version can be a simple number such as version 1, 2, 3, and so on, or it can be any alphanumeric string.

See "Importing an APX into SA - apxtool import" on page 112 and "Setting the current version of an APX - apxtool setcurrent" on page 115 for more information.

File structure

To SA, an APX is just a set of files and directories that conform to the contract of the APX type (Program APX or Web APX) such that the APX runtime can properly execute it. For example, a Web APX may need an index.html file or an index.php file. A Program APX may require a shell command with the same name as the APX.

For more information on the files in an APX, see "APX files" on page 116.

OGFS integration

The APX infrastructure depends on the OGFS to manage user sessions and to expose various parts of the APX in the SA file system. The following sections describe how APX is integrated into the OGFS and its various applications.

APX Executable Directory

Program APXs are treated as executable programs in the Global Shell, OGSH. These APXs are exposed as an executable command in the OGSH. This allows a shell user to invoke the APX as if running a shell command.

The APX executable directory has the following format:

```
/opsw/apx/bin/{apx_name}
```

where `apx_name` is the name of the APX. Running `apx_name` in `/opsw/apx/bin/{apx_name}`invokes the current version of `apx_name`.

APX Runtime Directory

The APX Runtime directory is used by the APX runtime to support execution of an APX. The APX Runtime directory must have access to the APX source. In addition, users who have developer privileges and have read permission to an APX can also access the APX. The APX Runtime directory is not available for non-APX developers in the Global Shell.

The APX Runtime directory references the source of the current version of an APX. It has the format:

```
/opsw/apx/runtime/{apx_type}/{apx_name}
```

where `apx_type` can be `script` or `web`.

APX Interfaces - Defining categories of APX Extensions

APX interfaces enable you to create named categories of APXs and to find all the APXs of a given category. An interface is the name of the category. For example, you could create a category of APXs that all take a certain set of input parameters and produces a certain type of output data. Or you could create a category of APXs that all perform a specific set of operations.

You can also create an APX or an external application that gets the names of all APXs of the desired category and executes them. Or the APX or application could just present the list of APXs of the desired category and let the user select one to execute.

An APX interface is a name that defines an informal contract between the caller of an APX and the APX.

- An APX that **defines an interface name** creates a category of APX with that name.
- An APX that **implements an interface** declares itself to be an APX of that category.

A sample interface

SA provides an interface named RightClickToRun. This interface defines a category of APX that takes one or more devices as input parameters and runs against those devices. In addition, the SA Client displays all APXs that implement this interface in the **Actions** > **Run Extension** menu, which allows users to select one or more devices and run these APXs against the selected devices. For more information on this interface, see "RightClickToRun interface" on page 105.

Defining an interface

An APX interface defines the name of a category of APXs. All APXs that implement the interface belong to the category and must adhere to the conventions of the interface. To create a new category, you make your APX “define” the interface.

To make your APX define an interface, perform the following steps:

1. Create the APX with the `apxtool new` command. For details on this command, see "Creating a new APX - apxtool new" on page 109.
2. Locate the files of your new APX and open the file named `interfaces` in a text editor. The interfaces file is located in the APX-INF directory of your APX directory.
3. At the end of the `interfaces` file, add three lines for:
   - The name of the interface section in the file. This is the unique name of the interface.
   - The display name of the interface.
   - A description of the interface.

   For example, the following shows the interface section name, the display name and the description of the interface named “com.hpe.sa.MyNewInterface”:

   ```
   [com.hpe.sa.MyNewInterface]
   name=MyNewInterface
   description=”This is a simple interface for testing purposes.”
   ```
4. Save your changes and close the file.
5. Import your modified APX into SA with the `apxtool import` command. For details on this command, see "Importing an APX into SA - apxtool import" on page 112.

To upgrade an existing APX to define an interface you must create the `interfaces` file and add your interfaces as described above.

Implementing an interface

An APX interface specifies a category of APX that adheres to the conventions of the interface. To specify that your APX belongs to a category, you make your APX “implement” the interface. To make your APX implement an interface, perform the following steps.

1. Create the APX with the `apxtool new` command. For details on this command, see "Creating a new APX - apxtool new" on page 109.
2. Locate the files of your new APX and open the file named `apx.cfg` in a text editor.

3. Locate the section in your `apx.cfg` file that discusses the “Implementing” section. This section briefly describes how to specify the interfaces that your APX implements.
4. Locate the following lines in the file `apx.cfg`:

   ```
   [Implementing]
   interfaces=
   ```

5. Modify the `interfaces=` line and add the name of your interface at the end of the line. For example, if your APX implements the interface named “com.hpe.sa.MyNewInterface”, the `apx.cfg` file would contain the following lines:

   ```
   [Implementing]
   interfaces=com.hpe.sa.MyNewInterface
   ```

   To implement more than one interface, add them to the interfaces line separated by colon, as follows:

   ```
   [Implementing]
   interfaces=com.hpe.sa.MyNewInterface:com.hpe.sa.AnotherInterface
   ```

6. Save your changes and close the file `apx.cfg`.
7. Import your modified APX into SA with the `apxtool import` command. For details on this command, see "Importing an APX into SA - apxtool import" on page 112.

You must set the current version of the APX to see the implemented interfaces when viewing the APX in the SA Client or with the `apxtool query` command. For more information, see "Setting the current version of an APX - apxtool setcurrent" on page 115.

To upgrade an existing APX to use an interface you must add your interfaces to your existing `apx.cfg` file as described above.

RightClickToRun interface

SA provides an interface you can use with your APXs named `com.hpe.client.server.RightClickToRun`. This interface works only with program APXs, not with web APXs. Use this interface when you want your APX to do all of the following:

- Take one or more devices as input parameters to the APX. APXs that implement this interface must take “`-d <device id>`” as an input argument.

- Appear in the **Actions > Run Extension >Select Extension...** window.
- Appear in the **Actions > Run Extension** menu of the SA Client. APXs appear in this menu after they have been run once using the **Actions > Run Extension >Select Extension...** menu.

To execute an APX from the **Actions > Run Extension** menu, the user must have execute permission on the APX. Any APX the user does not have permission to execute will not appear under this menu item. For information on permissions, see the Server Automation Administration Guide on the HPE SSO portal.

The RightClickToRun interface lets users select one or more devices in the SA Client and run your APX against those devices.

When you select the **Actions > Run Extension** menu item, the SA Client displays all of the program APXs that implement the interface com.hpe.client.server.RightClickToRun. When you select an APX, it is run against all the selected servers. The APX will be invoked once for each selected server.

For instructions on making your APX implement this interface, see "Implementing an interface" on page 104. For details on using an APX that implements this interface, see the Running SA Extensions section in the Server Automation Using Guide on the HPE SSO portal.

CoreAffinity interface

SA provides an interface that you can use with your APXs named ‘com.hpe.client.server.CoreAffinity’. You can use this interface when you want to run your APX in CoreAffinity mode.

CoreAffinity mode only applies when you have a mesh with at least two SA cores. When this mode is enabled for each target server, the APX is executed on the SA core to which this target server is registered, regardless of where the actual job was started.

For example:

- You have a mesh with two cores, core A and core B
- You start an APX job from core A on two target servers MA (registered to core A) and MB (registered to core B)

In core affinity mode this job runs the APX for MA on core A and MB on core B. If CoreAffinity is disabled then both executions will be done on core A (because that is where the job started).

For instructions on making your APX implement CoreAffinity interface, see "Implementing an interface" on page 104.

Using the Interface API

You can use the SA API to integrate your own applications with SA and APXs. Your application can determine all the APXs that implement a particular interface by using the interface named APXInterfaceService in the package named com.opsware.apx in the SA API. "API Documentation and the Twister" on page 37 on using the SA API.

apxtool command

Use the apxtool command in an OGFS session to create and manage APXs. The apxtool command is available in the Global Shell in the directory `/opsw/bin/apxtool`.

For a tutorial on how to use the apxtool to create a web APX, see "Tutorial: Creating a Web Application APX" on page 121.

Syntax of apxtool

Invoke the APX tool from the OGFS command line as follows:

```
apxtool [-h | --help] {function} arguments
```

To obtain a complete list of commands and arguments supported by the APX tool, run apxtool from an OGSH command line with no arguments.

The APX Tool supports the following major functions:

APX tool functions

Function	Usage
`new`	Creates a new APX source directory and a new set of template files in the OGFS. Optionally registers the APX into SA. Registering assigns an APX ID and makes the name of your APX available to others (with appropriate permissions) using SA. See "Creating a new APX - apxtool new" on page 109 for more information.
`import`	Imports your APX files into the SA Library and creates a new version of your APX. Optionally registers the APX into SA. Registering assigns an APX ID and makes the name of your APX available to others (with appropriate permissions) using SA. See "Importing an APX into SA - apxtool import" on page 112 for more information.

APX tool functions, continued

Function	Usage
`setcurrent`	Sets the current version of an APX in the SA Library. You can have multiple versions of an APX in SA, but only the current version can be executed. See "Setting the current version of an APX - apxtool setcurrent" on page 115 for more information.
`query`	Displays information about an APX. See "Querying APX information - apxtool query" on page 113 for more information.
`export`	Copies all of an APXs files from the SA Library to a separate set of files.
`delete`	Deletes an APX from the SA Library.

Using short and long command options

Most of the options to the apxtool command accept a short form or a long form.

- The short form is a single hyphen and a character, for example, “`-t`“ and “`-v`”.
- The long format is two hyphens followed by a word, for example, “`--type`“ and “`--view`“.

Some options require an argument following the option. For example, “`-t webapp`“ and “`-t details`“. Arguments can be specified in one of four formats, which are all equivalent. To illustrate, the following commands are equivalent and produce the same results:

```
apxtool query -t webapp
apxtool query -twebapp
apxtool query -tw
apxtool query --type webapp
apxtool query --type=webapp
```

Some options only require typing a minimum number of characters, enough to identify the option argument. For example, in the query function, the `--view` option requires argument “`list`“, “`details`“, “`versions`“. The following commands produce the same result:

```
apxtool query --view=details
apxtool query --view=d
apxtool query -vdetails
apxtool query -vd
```

Creating a new APX - apxtool new

You can use the APX tool to create a new APX and optionally register the name of the APX into SA. This command creates a set of template files for an APX that you can modify. For information on the files that make up an APX, see "APX files" on page 116.

Usage

```
apxtool new [options] {src_dir}
```

where the `src_dir` argument specifies the directory where the template files of the new APX are to be created. If this argument is omitted, the template files are placed into the current directory.

The following table lists the options for creating a new APX:

Options for apxtool new

Option	Usage
`-h, --help`	Show this help message and exit.

`-t <type>`
`--type=<type>`
	(**Required**) The APX type. Valid values are: `script` or `webapp`. For example, -ts for script APX, -tw for web APX. (A script APX is also known as a program APX.)

`-u <unique name>`
`--uniquename=<unique name>`
	(**Required**) The unique name of the APX. A unique name is a dot separated name that conforms to file system format. It must have at least one dot. Valid characters are: `[a-zA-Z0-9_.]`.
	Example: `com.hpe.sa.security.scan_ports`

`-n <name>`
`--name=<name>`
	(**Optional**) The display name of the APX in a folder. If a name is not specified, but a unique name is specified, the last part of the APX unique name is used as the display name. Note that this name must be unique within the specified folder.
	For example, if the unique name were com.hpe.sa.MyWebExt, the default display name would be MyWebExt.

`-d <description>`
`--description=<description>`
	(**Required**) A brief description of an APX. If the description is a filename with the extension .txt, the file is assumed to be a text file and its content is used as the APX description.

Options for apxtool new, continued

Option	Usage
`-r` `--register`	(**Optional**) Registers the name of the APX into the system. If you specify this option, you must also specify `-f` or `--folder`. If you do not specify -r and -f with `apxtool new`, you must use -f with `apxtool import`.

`-f <path>`
`--folder=<path>`
	(**Optional**) The SA folder path where the APX will be registered. This can be a full path, partial path, absolute path, or relative path, as long as it can uniquely identify a specific folder. This option is only needed if `-r` or `--register` is used.
	If you do not specify -r and -f with `apxtool new`, you must use -f with `apxtool import`.

`-Q, --quiet`	(**Optional**) Suppresses output messages.
`-F, --force`	(**Optional**) Suppresses confirmation prompts.

Deleting an APX - apxtool delete

You can use the APX tool to delete an existing APX from the SA library.

Usage

```
apxtool delete [options]
```

The following table lists the options for deleting an APX:

Options for apxtool delete

Option	Usage
`-h` `--help`	Show this help message and exit.

`-t <type>`
`--type=<type>`
	(**Required**) APX type. Valid values are: `script` or `webapp`. For example -ts for script.

`--id=<APX id>`	(**Optional**) The object identifier of the desired APX.
`-u <unique_name>`	(**Optional**) The unique name of the APX. A unique name is a dot separated name that conforms to file system format. It must have at least one dot.

Options for apxtool delete, continued

Option	Usage
`--uniquename=<unique_name>`	Valid characters are: `[a-zA-Z0-9_.]`. **Example**: `com.hpe.sa.security.scan_ports`
`-n <name>, --name=<name>`	(**Optional**) APX display name in a folder.
`-f <path>, --folder=<path>`	(**Optional**) SA folder path. Path can be a full path, partial path, absolute, or relative, as long as it can uniquely identify a specific folder.
`-Q, --quiet`	(**Optional**) Suppresses output messages.
`-F, --force`	(**Optional**) Suppresses confirmation prompts.

Exporting an APX from SA - apxtool export

You can use the APX tool to export an APX. Export downloads a specific version of an APX source archive file and places the files into a directory or into a .zip archive file.

Usage

```
apxtool export [options] {target_dir}
```

where the argument target_dir is the directory into which the APX source archive file is copied or into which the APX source archive content is expanded, depending on whether or not the --archive option is specified. If omitted, the current directory is used.

The following table lists the options for exporting an APX.

Options for apxtool export

Option	Usage
`-h, --help`	Show this help message and exit.
`-t <type>, --type=<type>`	(**Required**) APX type. Valid values are: `script` or `webapp`. For example, `-ts` for script.
`--id=<APX id>`	(**Optional**) The object identifier of the desired APX.
`-u <unique_name>,`	(**Optional**) The unique name of the APX. A unique name is a dot

Options for apxtool export, continued

Option	Usage
`--uniquename=<unique_name>`	separated name that conforms to file system format. It must have at least one dot. Valid characters are: `[a-zA-Z0-9_.]`. **Example**: `com.hpe.sa.security.scan_ports`
`-n <name>, --name=<name>`	(**Optional**) APX display name in a folder.
`-f <path>, --folder=<path>`	(**Optional**) SA folder path. Path can be a full path, partial path, absolute, or relative, as long as it can uniquely identify a specific folder.
`-v v<ersion_string>, --version=<version_string>`	(**Optional**) This option specifies which APX version to download. If omitted, the current version is downloaded.
`-a, --archive`	If specified, export the APX source in its original source archive as a ZIP or JAR file.
`-Q, --quiet`	(**Optional**) Suppresses output messages.
`-F, --force`	(**Optional**) Suppresses confirmation prompts.

Importing an APX into SA - apxtool import

You can use the APX Tool to import APXs. Import publishes a new version of an APX and optionally sets this version as the current version. If the APX was has not been registered yet, this command also registers the APX.

Only the current version of an APX can be run. If you do not set the current version, the APX will not be runnable. You can set the current version with either `apxtool import` or with `apxtool setcurrent`. See "Setting the current version of an APX - apxtool setcurrent" on page 115 for more information.

Usage

```
apxtool import [options] {apx_src}
```

where `apx_src` can be an archived APX source file with extension .zip or .jar or it can be the name of a directory containing the APX files to be published. apx_src may be a relative or absolute path. If omitted, the current directory is used. The specified directory or archive file must contain the directory APX-INF.

The following table lists the options that are available when importing an APX:

Options for apxtool import

Option	Usage
`-h, --help`	Show this help message and exit.
`-c, --setcurrent`	If specified, set the newly published version as the current version of an APX.
`--version=<version_string>`	The new version of this APX. This option must not be used if version_string is already specified in apx.cfg. If no version is specified, one will be assigned automatically.
`-f <path>, --folder=<path>`	(**Optional**) SA folder path. Path can be a full path, partial path, absolute, or relative, as long as it can uniquely identify a specific folder. If you did not specify -r and -f with `apxtool new`, you must use -r with `apxtool import`.
`-Q, --quiet`	(**Optional**) Suppresses output messages.
`-F, --force`	(**Optional**) Suppresses confirmation prompts.

Querying APX information - apxtool query

You can use the APX Tool to get and view APX information. You can specify additional options to limit resulting APXs. Multiple occurrences of the same option form a logical OR expression. If no matching result is found, this command returns exit code 100.

Usage

```
apxtool query [options]
```

The following table lists the options that are available when querying APX information:

Options for apxtool query

Option	Usage
`-h, --help`	Show this help message and exit.
`-v <view>, --view=<view>`	(**Optional**) Select one of the predefined views of the query results. Choices are `list` (default), `details`, and `versions`. `–v list` is a single line representation of APX basic information presented in tabular format. `–v details` is a multiple line representation of APX information. `–v versions` lists all APX versions. You would only need to specify enough characters for the view type; for example, -vd, is the same as -v details. If the versions layout is selected, the query must result in a single APX object.
`-t <type>, --type=<type>`	(**Optional**) Specifies the type of APX to display. Valid values are: script or webapp or interface. The default is to display all types. `-t script` displays all script APXs. `-t webapp` displays all web APXs. `-t interface` displays all APXs that define one or more interfaces. For example, `apxtool query` -ts displays all the script APXs.
`--id=<APX id>`	(**Optional**) The object identifier of the desired APX.
`-u <unique_name> --uniquename=<unique_name>`	(**Optional**) The unique name of the APX. A unique name is a dot separated name that conforms to file system format. It must have at least one dot. Valid characters are: [a-zA-Z0-9_.]. **Example**: `com.hpe.sa.security.scan_ports`
`-n <name>, --name=<name>`	(**Optional**) APX display name in a folder.
`-f <path>, --folder=<path>`	(**Optional**) SA folder path. Path can be a full path, partial path, absolute, or relative, as long as it can uniquely identify a specific folder.
`--current`	(**Optional)** if specified, only query APX objects that have a current version set.
`--format=<format_string>`	(**Optional**) This advanced option allows you to specify custom display formatting for an APX listing. `format_string` is a string containing embedded tag names that are substituted with values at display time. Tag names must have a format of `%(tag_name)`. Use the format string “`__show_tags__`” to display a list of all the

Options for apxtool query, continued

Option	Usage
	supported tag names.
`--csv`	(**Optional**) Displays the output in comma-separated values format. Ignored if the `--format` option is specified.
`-Q, --quiet`	(**Optional**) Suppresses extraneous output messages.

Setting the current version of an APX - apxtool setcurrent

You can use the APX tool to set an APX version as the current version.

Only the current version of an APX can be run. If you do not set the current version, the APX will not be runnable. You can set the current version with either `apxtool import` or with `apxtool setcurrent`. See "Importing an APX into SA - apxtool import" on page 112 for more information.

Usage

```
apxtool setcurrent [options] {version_str}
```

where the argument version_str is required to uniquely identify an existing version of an APX.

The following table lists the options that are available when setting an APX version:

Options for apxtool setcurrent

Option	Usage
`-h, --help`	Show this help message and exit.
`-t <type>, --type=<type>`	(**Required**) APX type. Valid values are: `script`, `webapp`. For example, -ts for script.
`--id=<APX id>`	(**Optional**) The object identifier of the desired APX.
`-u <unique_name>, --uniquename=<unique_name>`	(**Optional**) APX unique name. A unique name is a dot separated name that conforms to file system format.It must have at least one dot. Valid characters are [a-zA-Z0-9_.].

Options for apxtool setcurrent, continued

Option	Usage
	Example: `com.hpe.sa.security.scan_ports`
`-n <name>, --name=<name>`	(**Optional**) APX display name in a folder.
`-f <path>, --folder=<path>`	(**Optional**) SA folder path. Path can be a full path, partial path, absolute, or relative, as long as it can uniquely identify a specific folder.
`-Q, --quiet`	(**Optional**) Suppresses output messages.
`-F, --force`	(**Optional**) Suppresses confirmation prompts.

Error handling

The APX tool command conforms to the standard POSIX convention and returns 0 on success and a non-zero value for other errors. The APX tool sends normal output to STDOUT and errors and warnings to STDERR. When an error occurs, the APX tool typically returns a descriptive message to STDERR.

Error conditions are typically categorized as shown in the following table:

APX Tool Error Conditions

Return Code	Description
0	Success
1	Syntax or usage error
2	Permission related error
3	User canceled operation
4	Runtime error

There may be other undocumented exit codes. The only guarantee is that if the exit code is 0, the command completed its operation successfully.

APX files

This section describes the template files created when you run the `apxtool new` command. The following table summarizes these files. The sections below describe some of the files in more detail.

APX files

File name	Description
`apx.cfg`	APX configuration file, contains metadata that fully describes the APX. See "The APX configuration file - apx.cfg" below for more information.
`apx.perm`	APX permissions file, specifies permission escalation rules. See "The APX permissions escalation configuration file - apx.perm" on the next page for more information.
`description.txt`	Text description of the APX. Specified with the `apxtool new -d` option. See "Creating a new APX - apxtool new" on page 109 for more information.
`interfaces`	APX interface definition file. Specifies the interfaces the APX defines or implements. See "APX Interfaces - Defining categories of APX Extensions" on page 103 for more information.
`usage.txt`	Text description of how to use the APX.
`run.sh`	For program APXs only, this file contains the executable code of the APX. This file contains the functionality of the program APX. For an example, see "Tutorial: Creating a program APX" on page 129 for more information.
`index.php`	For web APXs only, this file contains the PHP source code for the web APX. This file contains the functionality of the web APX. For an example, see "Tutorial: Creating a Web Application APX" on page 121 for more information.

The APX configuration file - apx.cfg

All APXs regardless of type must have a configuration file named `apx.cfg`. The `apxtool new` command creates a template of this file for you to modify. This file contains metadata that fully describes the APX. The apx.cfg uses a “key=value” format to define the properties of the APX. Multiple lines are joined together with a line continuation character, “\“.

The "APX configuration file attributes" below table describes common attributes for all APXs. APX type specific attributes are described in the corresponding APX type functional specifications. Note that some of the attributes may be extracted from the `apx.cfg` configuration file and managed in SA. For modifiable attributes such as the description, subsequent updates of the `apx.cfg` file will update the SA managed data accordingly.

To see an example `apx.cfg` file, run the `apxtool new` command and open the files it creates.

APX configuration file attributes

Attribute	Modifiable?	Description
`type`	No	The type of the APX, which must be either `webapp` or `script`. (Script

APX configuration file attributes, continued

Attribute	Modifiable?	Description
		APXs are also known as Program APXs.) Once created, you cannot change the APX type.
`name`	Yes	This is the APX display name and may contain multi-byte characters. This name can be changed at any time. This name will be listed in the SA Client APX folders.
`unique_name`	No	The unique name of the APX. This name will be used as the file name for the APX as it appears in the OGFS. This name together with the type forms a key that uniquely identifies an APX. Once created, the name cannot be changed. Since this name is used in the file system, it must conform to the file system naming specification. Generally, this name should be in ASCII.
version	Yes	The version string representing the current version of the APX. If the value begins with the string “auto:”, then SA will automatically manage the versions using an integer incremented for each new version.
`description`	Yes	A text description of what the APX does. You can alternatively use the file `description.txt` instead of this attribute.
`usage`	Yes	A text description describing how to use the APX. You can alternatively use the file `usage.txt` instead of this attribute.
interfaces	Yes	One or more interfaces the APX implements. Separate multiple interfaces with a colon (:) character.
command	Yes	The executable file the APX is to run when it is invoked.

The APX permissions escalation configuration file - apx.perm

Use the file apx.perm to specify permission escalation rules. If this file does not exist, or if it contains no escalation permissions, the APX will run with the user's default permissions.

When a new APX is created using the APX Tool’s **New** command, it generates certain default files, including a default apx.perm file, which by default has no escalation permissions defined. The default file does contain some commented out examples which an APX developer can use as templates.

There are three ways to specify escalations, described below.

- "No escalation" below
- "All permissions" below
- "With escalation" below

No escalation

The escalations attribute is not specified. The APX runtime uses the current user privilege to execute an APX. If an APX invokes privileged operation which a user does not have, APX execution will terminate with an error.

All permissions

This is a special privilege that temporarily grants all operation permissions to a user. It is intended for development or demo use only. This is a useful tool for speedy proof of concept, or demo, without worrying fine grain permission tuning. It is a poor choice for a production environment due to its lack of security.

To grant all permissions, edit file apx.perm with a macro that matches all features with wildcard characters. For example:

```
use_feature(name=”*”)
```

With escalation

Specify a list of predefined common operations in the apx.perm file. When executing the APX, the APX runtime temporarily grants these permissions to the APX. SA has a comprehensive list of feature and resource permissions. To simplify the task of escalating related feature, one can use wildcard characters to match groups of related features. For example:

```
@use_feature(name=”Application.*”)
```

Showing the progress of an APX

You can use the `apxprogress` command in your program APX to provide information about the progress of your APX. This is useful for program APXs that run for a long period of time when you want to give the user status on the progress of your APX.

You can use a web APX as a front-end to the program APX and display the progress in the web APX.

The apxprogress command

Use the `apxprogress` command to define the number of steps in the execution of a program APX and to record when each step has completed. This lets users of the APX know how far the APX has progressed and how much is remaining.

Syntax of apxprogress

```
apxprogress {option}...
```

Options to the apxprogress command

Option	Description
`-i <total number of steps>`	Specifies the total number of steps the APX takes to run. Use this option once at the beginning of the APX to specify the total number of steps the APX will take. You can use this option multiple times in an APX to increase the number steps. Each use increments the total number of steps by the specified value.
`-c <current step>`	Specifies the current step number. Call `apxprogress` with this option after each step in the APX code has completed.
`-m <message>`	Specifies a text message describing the status of the APX.
`-a <data>`	Specifies additional information the APX can make available about itself.
`-d`	Indicates debug mode. Displays the output of the command to stdout for debugging purposes.
`-h`	Displays help information about the apxprogress command.

Example shell script that uses apxprogress

The following shell script is part of a program APX that uses the `apxprogress` command. The APX defines a total of 100 steps and announces its current progress 100 times. Each time it also provides a message that includes the step number.

```
#!/bin/sh
######################################################################
# A simple shell script for a program APX that displays progress
# about itself.
# Author: <name>
######################################################################
echo "This is a simple APX that uses apxprogress."
totalsteps=100
apxprogress -i $totalsteps -c 1
for i in `seq $totalsteps`; do
apxprogress -c $i -m "APX is running, working on step $i" -d
sleep 10
done
```

Viewing APX progress

You can use the SA API method `JobService.getProgress()`to access the progress information about a running APX that calls the `apxprogress` command. For an example showing this method, see "Viewing the APX progress in the Twister interface" on page 135, which is part of the "Tutorial: Creating a program APX" on page 129.

Tutorial: Creating a Web Application APX

This tutorial demonstrates how to create, publish, and run a simple web application APX named `mywebapp`.

Running the default version of the APX created during this tutorial displays the output of the PHP command, `phpinfo`. Later the tutorial shows you how to modify the PHP code so that it displays a list of managed servers. Because the tutorial provides the source code, prior knowledge of PHP is not required.

Complete the following tasks in order.

1. "Setting permissions and creating the tutorial folder" below
2. "Creating a new web application" on the next page
3. "Importing the new web application into SA" on page 125
4. "Running the new web application" on page 126
5. "Modifying the web application" on page 127
6. "Running the modified web application" on page 128

Tutorial prerequisites

To complete this tutorial, you must have the following capabilities and environment:

- The ability to log on to SA as `admin` or as another member of the **Super Administrators** group. Logging on as `admin` enables you to set permissions.
- The ability to log on to SA as a user who belongs to the **Advanced Users** group.
- Advanced users have permission to create and run the web application. In the example commands shown in this tutorial, the name of this user is `jdoe`.
- An understanding of how to set client feature permissions in the SA Client.
- For more information about permissions, see the "User and Group Setup" section in the the SA 10.50 Administration Guide.
- An understanding of how to create folders in the SA Client
- For details on folders, see the SA 10.50 User Guide.
- An understanding of how to open a Global Shell session.
- An understanding of basic Unix commands such as `ls` and `cd`.
- Experience developing web applications that run on HTTP servers.

Setting permissions and creating the tutorial folder

1. Log on to the SA Client as a member of the **Advanced Users** group and create the following folder in the SA Library:

   ```
   /Dev/MyApp
   ```

Later in the tutorial, you will upload a web application into the MyApp folder. In the non-tutorial environment, the name of this folder is arbitrary. You can create or choose any other folder to contain your web applications.

2. Exit the SA Client.
3. Log on to the SA Client as `admin` and open the **Folder Properties** of the `MyApp` folder.
4. On the **Permissions** tab of **Folder Properties**, make sure that the **Advanced** Users group has the following permissions:
   - List Contents of Folder
   - Read Objects Within Folder
   - Write Objects Within Folder
   - Execute Objects Within Folder
5. Exit the SA Client.

Creating a new web application

1. Open a Global Shell session as an SA user who belongs to the **Advanced Users** group.
2. In your core's OGFS home directory, create a directory named `mywebapp` and then change to that directory:

   ```
   $ mkdir mywebapp
   $ cd mywebapp
   ```

   The web application files will be stored in the `mywebapp` directory.
3. Using the `apxtool new` command, create the directory structure and default files for the web application as shown below.

   ```
   $ pwd
   /home/jdoe/mywebapp
   $ ls
   $
   $ apxtool new -tw -d "This is my first app." \
   -u com.hpe.sa.jdoe.mywebapp
   Create source directory /home/jdoe/mywebapp/com.hpe.sa.jdoe.mywebapp? Y/N y
   Info: Successfully created APX 'mywebapp' source directory:
   ```

```
/home/jdoe/mywebapp.
```

The `-tw` option indicates that the APX type is a web application, `-d` specifies a description, and `-u` specifies a unique name for the application.

For more information about the `apxtool new` command options, see the online help:

```
$ apxtool new -h
```

4. Change directories into the new directory created by the `apxtool new` command and list the files there.

   ```
   $ pwd
   /home/jdoe/mywebapp
   $ cd com.hpe.sa.jdoe.mywebapp
   $ ls
   APX-INF cgi-bin css images index.php
   $ ls -R
   .:
   APX-INF cgi-bin css images index.php
   ./APX-INF:
   apx.cfg apx.perm description.txt interfaces usage.txt
   ./cgi-bin:
   ./css:
   hp_sa.css
   ./images:
   ```

5. Display the contents of the default `index.php` file:

   ```
   $ cat index.php
   <?php
   // Show information about PHP
   phpinfo();
   ?>
   ```

   As with other web applications, you can replace the `index.php` file with an `index.html` file. However, this tutorial uses the `index.php` file, which you will modify in a later section.

6. Examine some of the files in the `APX-INF` directory. For more information, see "APX files" on page 116.

   The `APX-INF` directory contains information that is specific to APX web applications. As shown by the following `cat` command, the `description.txt` file holds the text you specified with the `-d` option of `apxtool new`.

```
$ ls APX-INF/
description.txt apx.cfg apx.perm usage.txt
$ cat APX-INF/description.txt
This is my first app $
```

The following `grep` command shows some of the properties in `apx.cfg`, the APX configuration file. The values for `type` and `uniquename` result from the `-t` and `-u` options of the `apxtool new` command. For details on the APX configuration file, see "The APX configuration file - apx.cfg" on page 117.

```
$ grep "=" APX-INF/apx.cfg
type=webapp
name=mywebapp
unique_name=com.hpe.sa.jdoe.mywebapp
```

Importing the new web application into SA

Importing the web application performs the following actions:

- Installs the web application on an HTTP server within SA.
- Copies the web application to a folder that appears in the SA Library and in the Global Shell.
- Assigns a version number to the web application.

Enter the `apxtool import` command and respond to the prompts with `y`, as shown below. The `-f` option specifies the folder in the SA Library where the web application will be stored. The `-c` option sets the current version of the web application.

```
$ pwd
/home/jdoe/mywebapp/com.hpe.sa.jdoe.mywebapp
$
$ apxtool import -f "/Dev/MyApp" -c
APX source is not specified.
Do you want to publish current directory: /home/jdoe/mywebapp/
com.hpe.sa.jdoe.mywebapp? Y/N y
APX with unique name 'com.hpe.sa.jdoe.mywebapp' does not exist.
Register it into the system? Y/N y
Info: Successfully registered APX 'mywebapp' (310001) in folder ‘/Dev/
```

```
MyApp’.
Info: Successfully published a new version '1' for APX 'mywebapp'.
Info: Successfully set APX 'mywebapp'(310001) current version as '1'.
```

Running the new web application

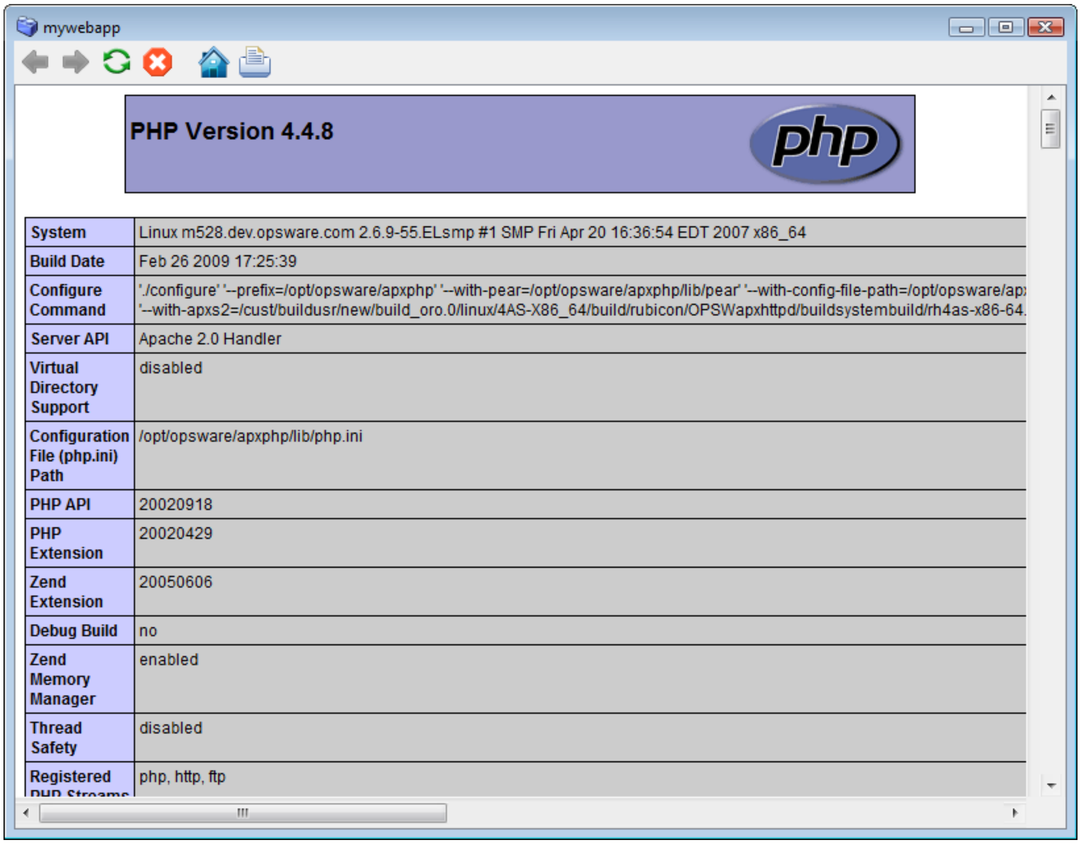
Now that you have published the web application, you are ready to run it from the SA Client, just as an end-user would.

1. Log on to the SA Client as a user who belongs to the **Advanced Users** group.
2. Select the Library tab and the By Type tab.
3. Navigate to the **Extensions > Web** node where you should see the `mywebapp` extension.

   If you do not see `mywebapp`, make sure that you have the necessary permissions as described in "Setting permissions and creating the tutorial folder" on page 122.
4. To run the web application, select `mywebapp`. and select the **Actions** > **Run** menu.

   The following figure appears. The web application displays the information generated by the `phpinfo` statement of the `index.php` file.

Web Application Version 1



PHP Version 4.4.8

System	Linux m528.dev.opsware.com 2.6.9-55.ELsmp #1 SMP Fri Apr 20 16:36:54 EDT 2007 x86_64
Build Date	Feb 26 2009 17:25:39
Configure Command	'./configure' '--prefix=/opt/opsware/apxphp' '--with-pear=/opt/opsware/apxphp/lib/pear' '--with-config-file-path=/opt/opsware/ap '--with-apxs2=/cust/buildusr/new/build_oro.0/linux/4AS-X86_64/build/rubicon/OPSWapxhttpd/buildsystembuild/rh4as-x86-64.
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/opt/opsware/apxphp/lib/php.ini
PHP API	20020918
PHP Extension	20020429
Zend Extension	20050606
Debug Build	no
Zend Memory Manager	enabled
Thread Safety	disabled
Registered PHP Streams	php, http, ftp

Modifying the web application

Running the default `index.php` file is a good way to check your development environment, but it does not take advantage of SA functionality. In this section, you modify the `index.php` file so that it lists the names of servers managed by SA.

1. In the Global Shell session, locate the `index.php` file of the web application.

   ```
   $ cd /home/jdoe/mywebapp/com.hpe.sa.jdoe.mywebapp
   $ ls
   APX-INF cgi-bin css images index.php
   ```

2. Open the `index.php` file in a text editor such as `vi`.
3. Replace the contents of `index.php` with the following lines:

   ```
   <html>
   ```

```
<head>
<title>Servers</title>
</head>
<body>
<p>List of servers:</p>
<?php
passthru("ls /opsw/Server/@");
?>
</body>
</html>
```

The `passthru` statement above runs the `ls` command and passes `stdout` (without reinflates) back to the web page. The `ls` command lists the names of your managed servers as they appear in the OGFS.

4. Save the `index.php` file and exit the text editor.
5. Publish the modified web application.

   The following `apxtool import` command sets the current version to 2. The `-F` option suppresses the confirmation prompts.

```
$ apxtool import -f "/home/jdoe/mywebapp/com.hpe.sa.jdoe.mywebapp" \
-c --version=2 -F
Info: Successfully published a new version '2' for APX 'mywebapp'
Info: Successfully set APX 'mywebapp'(310001) current version as '2'.
```

Running the modified web application

1. In the SA Client, use the **View> Refresh** menu to refresh the view of your web extensions, which should now contain version 2 of `mywebapp`.
2. Select `mywebapp` and select the **Actions > Run** menu. The output should be similar to the Web Application Version 1 except it displays the output of the PHP `passthru` statement and the OGSH `ls` statement, which lists all your managed servers. Note that the `passthru` statement removes the line feeds that separate the server names returned by the `ls` command.

Tutorial: Creating a program APX

This tutorial demonstrates how to create, publish, and run a simple program APX named myshellapp that runs a simple shell script. Later the tutorial shows you how to modify the shell script to call the `apxprogress` command and provide progress information. Because the tutorial provides the source code, prior knowledge of shell programming is not required.

Complete the following tasks in order.

- "Setting permissions and creating the tutorial folder" on the next page
- "Creating a new program APX" on page 131
- "Importing the new APX into SA" on page 133
- "Running the new APX" on page 133
- "Modifying the APX" on page 134
- "Running the modified APX" on page 135
- "Viewing the APX progress in the Twister interface" on page 135

Tutorial prerequisites

To complete this tutorial, you must have the following capabilities and environment:

- The ability to log on to SA as `admin` or as another member of the **Super Administrators** group. Logging on as `admin` enables you to set permissions.
- The ability to log on to SA as a user who belongs to the **Advanced Users** group.
- Advanced users have permission to create and run the web application. In the example commands shown in this tutorial, the name of this user is `jdoe`.
- An understanding of how to set client feature permissions in the SA Client.
- For more information about permissions, see the "User and Group Setup section" in the SA 10.50 Administration Guide.
- An understanding of how to create folders in the SA Client
- For details on folders, see the SA 10.50 User Guide.

- An understanding of how to open a Global Shell (OGSH) session and use the Global Shell.
- An understanding of basic Unix commands such as `ls` and `cd`.

Setting permissions and creating the tutorial folder

1. Log on to the SA Client as `admin` and open the **Folder Properties** of the `MyApp` folder.
2. On the **Permissions** tab of **Folder Properties**, make sure that the **Advanced** Users group has the following permissions:
   - List Contents of Folder
   - Read Objects Within Folder
   - Write Objects Within Folder
   - Execute Objects Within Folder
3. Exit the SA Client.

Creating a new program APX

1. Open a Global Shell session as an SA user who belongs to the **Advanced Users** group.
2. In your core’s OGFS home directory, create a directory named `myshellapp` and then change to that directory:

   ```
   $ mkdir myshellapp
   $ cd myshellapp
   ```

   The program APX files will be stored in the `myshellapp` directory.
3. Using the `apxtool new` command, create the directory structure and default files for the program APX as shown below.

   ```
   $ pwd
   /home/jdoe/myshellapp
   $ ls
   $
   $ apxtool new -ts -d "This is my first program APX." \
   -u com.hpe.sa.jdoe.myshellapp

   Create source directory under
   ‘/home/jdoe/myshellapp/com.hpe.sa.jdoe.myshellapp’ for APX ‘myshellapp’? Y/N y
   Info: Successfully created source directory
   ‘/home/jdoe/myshellapp/com.hpe.sa.jdoe.myshellapp for APX ‘myshellapp’.
   ```

   The `-ts` option indicates that the APX type is a program APX (also called a script APX), `-d` specifies a description, and `-u` specifies a unique name for the application.

   For more information about the `apxtool new` command options, see the online help:

   ```
   $ apxtool new -h
   ```
4. List the files created by the `apxtool new` command:

   ```
   $ pwd
   /home/jdoe/mywebapp
   $ ls
   com.hpe.sa.jdoe.myshellapp
   ```

```
$ cd com.hpe.sa.jdoe.myshellapp
$ pwd
/home/jdoe/myshellapp/com.hpe.sa.jdoe.myshellapp
$ ls -R
.:
APX-INF run.sh
./APX-INF:
apx.cfg apx.perm description.txt interfaces usage.txt
```

5. Display the contents of the default `run.sh` file:

```
$ cat run.sh
#!/bin/sh

######################################################################
# APX myshellapp
#
# Created by: jdoe
#
######################################################################
echo "This is APX myshellapp"
```

6. Examine some of the files in the `APX-INF` directory. For more information on these files see "APX files" on page 116.

   The `APX-INF` directory contains information that is specific to APXs. As shown by the following `cat` command, the `description.txt` file holds the text you specified with the `-d` option of `apxtool new`.

```
$ ls APX-INF/
apx.cfg apx.perm  description.txt  interfaces  usage.txt
$ cat APX-INF/description.txt
This is my first program APX.$
```

   The following `grep` command shows some of the properties in `apx.cfg`, the APX configuration file. The values for `type` and `uniquename` result from the `-t` and `-u` options of the `apxtool new` command. For details on the APX configuration file, see "The APX configuration file - apx.cfg" on page 117.

```
$ grep "=" APX-INF/apx.cfg
type=script
name=myshellapp
```

```
unique_name=com.hpe.sa.jdoe.myshellapp
command=run.sh
```

Importing the new APX into SA

Importing the APX performs the following actions:

- Copies the APX to a folder that appears in the SA Library.
- Assigns a version number to the APX.

Enter the `apxtool import` command and respond to the prompts with `y`, as shown below. The `-f` option specifies the folder in the SA Library where the web application will be stored. The `-c` option sets the current version of the web application.

```
$ pwd
/home/jdoe/myshellapp/com.hpe.sa.jdoe.myshellapp
$
$ apxtool import -f "/Dev/MyApp" -c
APX source is not specified.
Do you want to publish current directory: /home/jdoe/myshellapp/
com.hpe.sa.jdoe.myshellapp? Y/N y
APX with unique name 'com.hpe.sa.jdoe.myshellapp' does not exist.
Register it into the system? Y/N y
Info: Successfully registered APX 'myshellapp' (20001).
Info: Successfully published a new version '1' for APX 'myshellapp'
Info: Successfully set APX 'myshellapp'(20001) current version as '1'.
```

Now that you have published the APX, you are ready to run it from the SA Client, just as another SA user would.

Running the new APX

Now that you have published the APX, you are ready to run it from the SA Client.

1. Log on to the SA Client as a user who belongs to the **Advanced Users** group.
2. In the navigation pane, select the Library tab, then the By Type tab.
3. Open the Extensions node and select the Program node. This displays all the program APXs in the SA Library. You should see your APX there. If you do not see `myshellapp`, make sure that you

have the necessary permissions as described in "Setting permissions and creating the tutorial folder" on page 130.

4. Select your APX.
5. Select the **Actions > Run** menu item. This displays the Run Program Extension wizard.
6. Select the Next button.
7. Select the Start Job button.
8. When your APX finishes, select the status indicator to display details.
9. Select the Close button.

Modifying the APX

In this section, you modify the `run.sh` file and add calls to the `apxprogress` command to provide progress information.

1. In the Global Shell session, locate the `run.sh` file of the APX.

```
$ cd /home/jdoe/myshellapp/com.hpe.sa.jdoe.myshellapp
$ ls
APX-INF run.sh
```

2. Open the `run.sh` file in a text editor such as `vi`.
3. Replace the contents of `run.sh` with the following lines:

```
echo "This is a simple APX that uses apxprogress."

totalsteps=100
apxprogress -i $totalsteps -c 1
for i in `seq $totalsteps`; do
        apxprogress -c $i -m "myshellapx is running, working on step $i" #-d
        sleep 10
done
```

These apxprogress commands specify that the APX has 100 steps and it calls apxprogress 100 times, once for each step, waiting ten seconds between calls. For more information, see "Showing the progress of an APX" on page 120.

For debugging, you can change “#-d” to “-d” and run the shell script manually to display the output from the apxprogress commands on stdout.

4. Save the run.sh file and exit the text editor.
5. Publish the modified APX.

   The following apxtool import command loads the new version of the APX and sets the current version to 2. The -F option suppresses the confirmation prompts.

```
$ apxtool import -f "/home/jdoe/myshellapp" \
-c --version=2 -F
Info: Successfully published a new version '2' for APX 'myshellapp'
Info: Successfully set APX 'myshellapp'(20001) current version as '2'.
```

Running the modified APX

Now that you have modified and republished the APX, run it from the SA Client as before.

1. In the SA Client, use the **View >Refresh** menu to refresh the view of the program extensions, which should now show version 2 of myshellapp.
2. Select your APX.
3. Select the **Actions** > **Run** menu item. This displays the Run Program Extension wizard.
4. Select the Next button.
5. Select the Start Job button.

Viewing the APX progress in the Twister interface

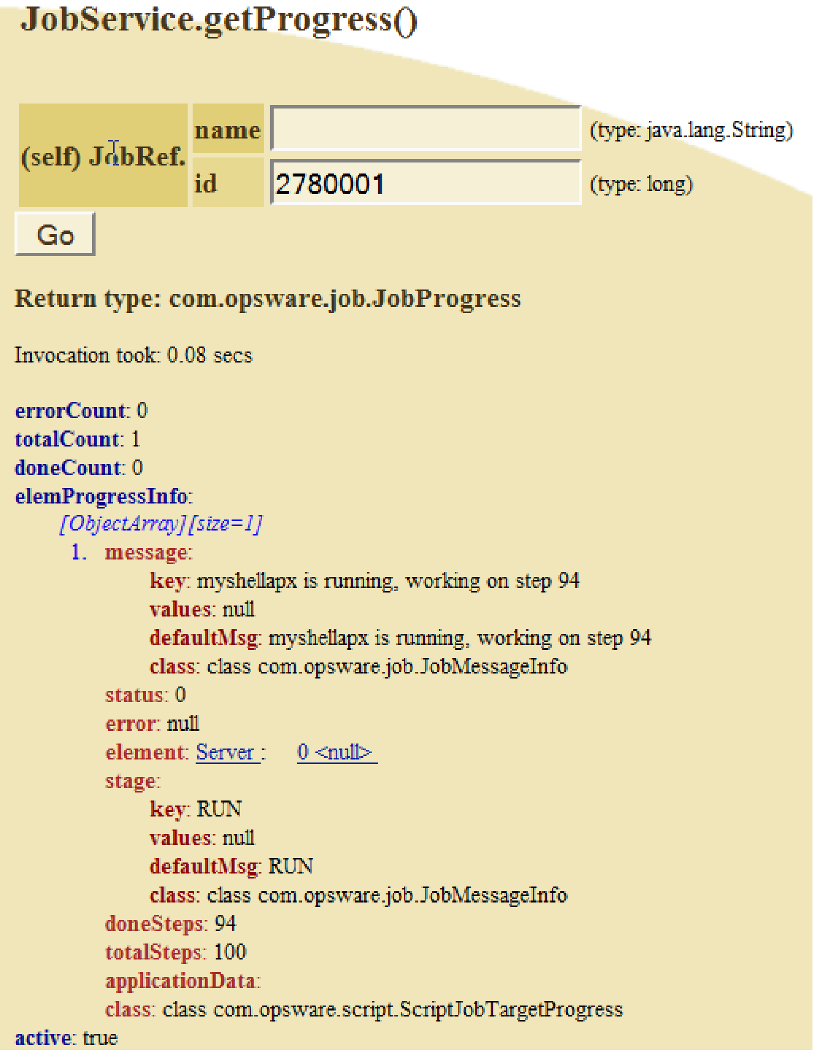
The apxprogress commands report the progress of the running APX. You can obtain this progress information by calling the API method JobService.getProgress(). This section shows you how to run this method from the Twister interface. For more information on the Twister interface to the SA API, see "API Documentation and the Twister" on page 37.

1. In the SA Client, select the Jobs and Sessions tab.
2. Locate your APX in the list of jobs.
3. Note the Job ID number of your APX job. You will use this in a later step.
4. Run the SA Twist interface by entering the following URL into a web browser:

   ```
   https://<core_host>:1032
   ```

   where <*core_host*> is the IP address or host name of your SA core server. This displays the Twist interface to the SA API in the web browser.
5. Select the “Twister” link. This displays the Twister interface to the SA API where you can get complete information about API interfaces, packages and methods and where you can run methods.
6. Locate and select the `JobService` interface, which is in the `com.opsware.job` package.
7. Scroll down and locate the `getProgress()` method.
8. Select the Try It button just above the `getProgress()` method.
9. Enter your SA credentials.
10. Select the Login button.
11. In the “id” field, enter the job number of your running APX, from step 3 above.
12. Select the **Go** button. This calls the `getProgress()` method and displays the current progress information about your APX from the `apxprogress` command, as shown below. Notice that the total number of steps is 100 and the number of completed steps is 94 in this snapshot. For more information on the output from the `getProgress()` method, see the Javadocs documentation by

selecting the `getProgress()` method in the navigation pane of the Twister web browser.



JobService.getProgress()

(self) JobRef.	name		(type: java.lang.String)
	id	2780001	(type: long)

Return type: com.opsware.job.JobProgress

Invocation took: 0.08 secs

```
errorCount: 0
totalCount: 1
doneCount: 0
elemProgressInfo:
    [ObjectArray][size=1]
    1. message:
           key: myshellapx is running, working on step 94
           values: null
           defaultMsg: myshellapx is running, working on step 94
           class: class com.opsware.job.JobMessageInfo
       status: 0
       error: null
       element: Server :    0 <null>
       stage:
           key: RUN
           values: null
           defaultMsg: RUN
           class: class com.opsware.job.JobMessageInfo
       doneSteps: 94
       totalSteps: 100
       applicationData:
       class: class com.opsware.script.ScriptJobTargetProgress
active: true
```

Agent Tools

Introduction to Agent Tools

Agent Tools is a suite of shell scripts, batch files, and Python scripts specifically designed to retrieve and modify information about Managed Servers. The information is retrieved from and modified in the SA database.

Using the scripts, you can retrieve and modify such data as custom fields, customer assignments, custom attributes, and more. Given this ability, you can automate many procedures that in the past had to be accomplished on a server-by-server basis.

In addition, you can incorporate the information the scripts retrieve into customized scripts of your own design. Since information such as customer assignment and custom attributes varies from managed server to managed server, the ability to retrieve and use this information *on-the-fly* in customized scripts can be very useful.

For example:

- You may have a script that handles post-installation configuration for a certain application that must be able to discover the Facility name in which the server is registered. Agent Tools provides a script to get the Facility name and insert it into your post-installation script without manual intervention.
- When installing a monitoring agent, a post-installation script must modify a configuration file to include the IP address of the monitoring server in that particular facility. Agent Tools provides a script to discover the monitoring server’s IP address by reading a custom attribute on the Core so that it can be inserted into the configuration file.
- A DSE can be written to retrieve the EEPROM version from many servers and store that information as a custom attribute or custom field.

Some other uses of Agent Tools scripts include:

- Gathering information from an SA Core during software installation for use in configuration.
- Storing metadata from managed servers in the SA database while executing a DSE, Global Shell script, or software installation.
- Retrieving custom attribute information for Managed Servers.

Installation requirements

The Agent Tools suite has the following requirements:

Operating System support

Agent Tools supports the operating systems supported by the SA Managed Servers. For a list of supported operating systems, See the Server Automation Installation Guide.

Security, access control, and authentication

Agent Tools must be run as the *root user* on UNIX/Linux systems or as the *Administrator* on Windows systems. Agent Tools use the Server Agent's certificate to connect to the Web Services Data Access Engine (twist) which is pyTwist’s default behavior, and is granted the privileges that the Web Services Data Access Engine gives to the Agent. This typically applies to read/write privileges on the server from which Agent Tools is run, therefore, no user authentication is required.

An exception is the `set_customer` script. You must have read access to a customer to be able to associate a server with that customer. Agent certificates do not have read access to other customers, therefore the user must authenticate when running this script.

Running Agent Tools scripts on Windows is not supported when UAC (User Access Control) is enabled.

Other requirements

- Access privileges to pyTwist
- Access privileges to the SA API
- Installed Python 2.4 (shipped with the Server Agent)

Installation

Agent Tools is installed in the Core during the normal HPE SA Installer Core installation process. However, you must also install Agent Tools on your Managed Servers to make it available on those servers. This section describes that process.

Agent Tools is installed on Managed Servers as a set of executable scripts. Depending on your operating system, these will be shell or batch scripts and Python scripts which are called by the shell and batch scripts. You can run these scripts from a managed server to retrieve and modify information in the SA Core. These scripts can be run manually or called from package installation scripts, DSEs, Global Shell scripts, and so on.

Agent Tools is included as part of the Python SA API Access (pyTwist) software policy. This policy is located in the directory:

```
/Opsware/Tools/Python Opsware API Access
```

Manually installing Agent Tools

To install Agent Tools on a Managed Server:

1. Launch the SA Client.
2. Go to the **Managed Servers** list and select the Managed Server(s) on which you want to install Agent Tools.
3. Right click and select **Install Software**.
4. Select the **Python Opsware API Access** software Policy.
5. The Software Policy installation wizard will guide you through the rest of the process.

Installing Agent Tools when installing an Agent

Alternatively, you can specify the Python SA API Access software Policy ID and specify that it be remediated during Agent installation. For information about Agent installation, see Administer.

Upgrading Agent Tools

Since Agent Tools is provided as a software policy (part of the pyTwist software policy), you can upgrade to newer versions of Agent Tools by performing a remediation after upgrading the core.

When the SA core is upgraded, the Python SA API Access software policy is also updated; any old versions of Agent Tools are removed and new versions are attached to the policy. After the SA Core upgrade (during which Agent Tools will be automatically upgraded as part of the core upgrade), you can then upgrade Agent Tools on the Managed Servers by performing the following tasks:

1. Select the managed servers that have had Agent Tools installed. You can see a list of the servers and groups attached to the Python SA API Access software policy by opening the policy itself.
2. Right click on the selected servers and choose **Remediate**.
3. Select the **Python Opsware API Access** software policy.
4. The old versions of the pyTwist and Agent Tools packages are removed, and the new versions are installed.

Data migration

Since Agent Tools keeps no persistent data on the managed server, there's no requirement for data migration or preservation.

Agent Tools scripts

Usage

```
<scriptname>.py|bat|sh --arguments
```

Agent Tool scripts

Script	Function
`get_all_cust_attr`	Retrieves all custom attributes for a server record. Usage: `get_all_cust_attr.py [--localonly] [--mode=python|shell|pretty]` The mode determines the format for the output (such as Python dictionary, shell statements, etc.). Pretty is the default. **Note:** Shell mode does not work when there are multi-line custom attributes.
`get_cust_attr`	Retrieves the value of a single custom attribute. **Usage**: `get_cust_attr.py [--localonly] <custom attribute name>`

`set_cust_attr`	Sets the value of a single custom attribute on the server.
	Usage: set_cust_attr.py
	`<custom attribute name>`
	`<custom attribute value>|--valuefile`
	`<path to file with value in it>`

`del_cust_attr`	Deletes a custom attribute from the server's record in the database. **Usage**: `del_cust_attr.py <custom attribute name>`
`get_cust_field`	Retrieves the value of a single custom field. **Usage**: `get_cust_field.py <custom field name>`
`set_cust_field`	Sets the value of a single custom field on the server. **Usage**: set_cust_field.py <custom field name> <custom field `value>|--valuefile <path to file with value in it>`
`get_customer`	Retrieves the customer name that the server is associated with.

Agent Tool scripts, continued

Script	Function
	Usage: `./get_customer.py`
`set_ customer`	Sets the customer name that the server is associated with. **Usage**: set_customer.py <customer name>
`get_ facility`	Retrieves the name of the Facility that the server is associated with. **Usage**: `./get_facility.py`
`get_info`	Prints out all fields for a server (in a format similar to the server's info file in OGSH). **Usage**: `get_info.py`

`get_ history`	Prints out server specific events.
	Usage:
	`get_history.py --startdate <start date in seconds since epoch>`
	`[--enddate <end date in seconds since epoch>]`
	`[--username <SAS user name>] [--password <SAS password>]`

`sub_ text_ file`	Reads in a text file, looks in the file for tokens/parameters, replaces them with the value of custom attributes, and prints the amended file to stdout. See below for more info on the expected file format. **Usage**: `sub_text_file.py [--localonly] <path to file with tokens in it>`

Formatting for the sub_text_file script

Text files passed to the `sub_text_file` script can have any content, however, the script looks for any lines with two @ characters and will treat the string between and including the @ character pairs as a token. You can have a single @ character on a line, it will be ignored, however a second @ character on the same line will cause any text between the two @ characters to be treated as a token.

The tokens are replaced with the value of the custom attribute specified between the @ signs. For example, the string @dns_server@, is replaced with the value of the custom attribute dns_server. If this custom attribute does not exist or its value is empty, the token is replaced with an empty string.

Take a text file that contains the entry:

```
IP: @monitoring_server_ip@
```

The script will output will look similar to the following:

```
IP: 82.159.202.117
```

Where IP is the value retrieved by `monitoring_server_ip`.

Output

The `sub_text_file` script outputs to stdout. You can redirect the output to a file if needed. You can also use a `.template` file stored in your zip file to format the output. For example:

```
$AGENTTOOLSPATH/sub_text_file.sh petstore_config.template > petstore_config.cfg
```

Sample Agent Tool scripts

The following are simple examples of using Agent Tools scripts.

UNIX/Linux

This example puts a message containing the name of the facility in the Message of the Day (MOTD) that users see when they log into the UNIX server.

```
. /etc/opt/opsware/pytwist/pytwist.conf
facility_name=`$AGENTTOOLSPATH/get_facility.sh`
echo "You have connected to a server in the $facility_name facility. For hardware
information on this server as stored in Opsware, run $AGENTTOOLSPATH/get_info.sh."
> /etc/motd
```

Windows

This Windows example puts a text file on all users' desktops with information about the server.

```
call "C:\Program Files\Common Files\Opsware\etc\pytwist\
pytwist_conf.bat"
```

```
call"%AGENTTOOLSPATH%\get_info.bat" > "%SYSTEMDRIVE%\Documents and Settings\All
Users\Desktop\server_info_from_Opsware.txt"
```

1. Do not hard code the path to Agent Tools Instead you must do the following:

   Source the PyTwist configuration file:

   UNIX:
   `./etc/opt/opsware/pytwist/pytwist.conf`

   Windows:

```
call
C:\Program Files\Common Files\Opsware\etc\pytwist
\pytwist_conf.bat
```

2. Use the environment variable:

   UNIX:
   ```
   $AGENTTOOLSPATH
   ```

   Windows:
   ```
   %AGENTTOOLSPATH%
   ```

   Using this method will prevent errors in your scripts should the path to Agent Tools change in future.

Microsoft Windows PowerShell - SA integration

Introduction to Microsoft Windows PowerShell

Windows PowerShell is an extensible command shell for system administrators and programmers, integrated with Microsoft's .Net 2.0 Framework Class Library. It uses the .NET common language runtime and the .NET Framework, and accepts and returns .NET objects. This enhances the tools and methods available to manage and configure of Windows.

Windows PowerShell provides numerous *cmdlets*, which are built into the shell and provide a wide range of functionality. Cmdlets can be used individually or in combination to perform more complex tasks.

Windows PowerShell not only enables access to a computer's file system, PowerShell *Providers* allow you to access data stores like the registry and digital signature certificate stores. A *Provider* is a software module that provides a uniform interface between a service and a data source.

Before you attempt to use the Windows PowerShell with SA, it is assumed that you are familiar with and comfortable using Microsoft Windows PowerShell. If you need background or instruction in using PowerShell, see *http://www.microsoft.com*.

> **Caution:** Because the included cmdlets can modify data on your managed servers, it is important that you have a solid understanding of Windows PowerShell and its use.

Windows PowerShell integration with SA

SA provides initial integration with Microsoft Windows PowerShell on managed servers running Windows. PowerShell is available from SA user interfaces and SA data is available from within the standard PowerShell environment or from within any PowerShell Runspace. A *PowerShell Runspace* is a hosting environment for the PowerShell runtime system.

The following PowerShell cmdlets are available with SA:

- Get-SASServer
- Set-SASServer
- Get-SASJob

SA also includes a PowerShell *SAS Provider* (a component that provides access to the objects in an SA core in a PowerShell environment).

Integrated PowerShell/SA cmdlets

The lists below and describes the integrated PowerShell/SA cmdlets included with SA.

PowerShell cmdlets

Cmdlet	Description	Arguments

`Get-SASServer`	Retrieves server data from specified server(s)	`-Credential <PSCredential>`
		`-Core <Hostname|IPAddress>`
		`-Name < ListOfHostnameFragments> |`
		`-Id <ListOfServerIDs>`

`Get-SASJob`	Retrieves data for specified jobs	`-Credential <PSCredential>`
		`-Core <Hostname|IPAddress>`
		`-JobFilter <ListOfJobIDs>`

`Set-SASServer`	Retrieves a list of managed servers	`-Credential <PSCredential>`
		`-Core <Hostname|IPAddress>`
		`-Server <ServerVO>`

> **Caution:** If the target core is running a minimum protocol version of TLSv1.x, the Powershell version (the bound underlying .NET Framework version) must support it. For more information see, https://msdn.microsoft.com/en-us/library/system.net.securityprotocoltype(v=vs.110).aspx.

Installation requirements

An MSI installer package containing the cmdlets and PowerShell SA Provider assemblies, configuration and setup files for installation on a System Administrator's Windows desktop.

Operating System support

- Windows Server 2003
- Windows Server 2008
- Windows Server 2008 R2 x64
- Windows Server 2012

Installation

To implement Microsoft Windows PowerShell/SA integration, you must perform the following tasks:

- Locate the Microsoft Windows PowerShell/SA Connector MSI package in the OCC **Library**>**Software Policies**.
- Run the MSI to install the assemblies that define the SA-specific cmdlets and SA Provider. The file `readme.rtf` provides last minute information. The Microsoft Windows PowerShell initialization script, profile.ps1 (similar to .bashrc) and a set of sample PowerShell scripts that show how to use PowerShell in an SA environment are also installed.

By default, the MSI installs the connector into `C:\Program Files\Opsware\PsSas`.

The file, `SAS-WSAPI.ps1`, describes accessing the WS-API directly from PowerShell, without the need for cmdlets.

Microsoft Windows PowerShell integration with SA features

Microsoft Windows PowerShell is available as an option in the following areas:

- "Remote access to managed servers" below
- "Audit and snapshots rules" below
- "DSE script integration" on the next page

Remote access to managed servers

From the SA Client, you can open a remote PowerShell session for any managed server (not available for a group of servers), as you would when opening a remote terminal.

1. Launch the SA Client.
2. From the Navigation pane, select **Devices**>**All Managed Servers**.
3. Select a Managed Server and open it.

   In the Device Explorer window, from the **Actions** menu, select **Launch Remote PowerShell**.

You cannot run a script that contains *WMI calls* while logged in to a remote PowerShell session. If you try to run a script containing WMI call, you will get an `Access Denied` error, even if you are a member of a group with the necessary permissions to run that script.

Audit and snapshots rules

Microsoft PowerShell is integrated with SA auditing. While configuring a custom script rule, Microsoft PowerShell scripts are now an option along with batch, Python 2 and Visual Basic. For details about audit, see the Server Automation Administration Guide on the HPE SSO portal.

DSE script integration

For Managed Servers, you can set up PowerShell scripts that call SA APIs using Pytwist so that end users can invoke the scripts as DSEs or ISM controls. For more information about writing scripts that invoke Pytwist APIs, see "Python API access with Pytwist" on page 77.

Sample sessions

This section provides four scenarios that demonstrate using Windows PowerShell/ SA integration.

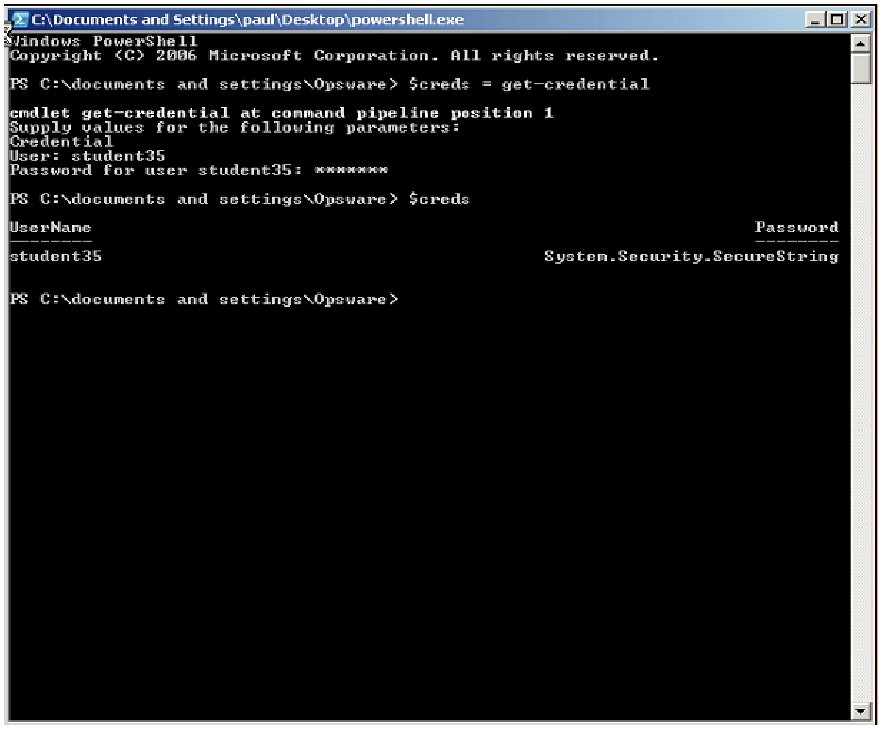
- "Scenario 1" below demonstrates extracting managed server data from an SA Core, modifying it, and writing it back to the core.
- "Scenario 2" on page 158 demonstrates exporting SA managed server data to an Excel spreadsheet using Windows PowerShell/SA integration.
- "Scenario 3" on page 160 demonstrates mounting the SA core as a Windows PowerShell PSdrive and navigating around the virtual file system.
- "Scenario 4" on page 163 demonstrates listing all the types of SA objects available to a Windows PowerShell environment.

Scenario 1

Authenticating to an SA Core, obtaining data about a managed server, modifying the data, and writing the data back to the SA Core.

1. Open a PowerShell prompt from the desktop icon.
2. Store the SA Core credentials securely in a PowerShell shell variable. See the following figure.

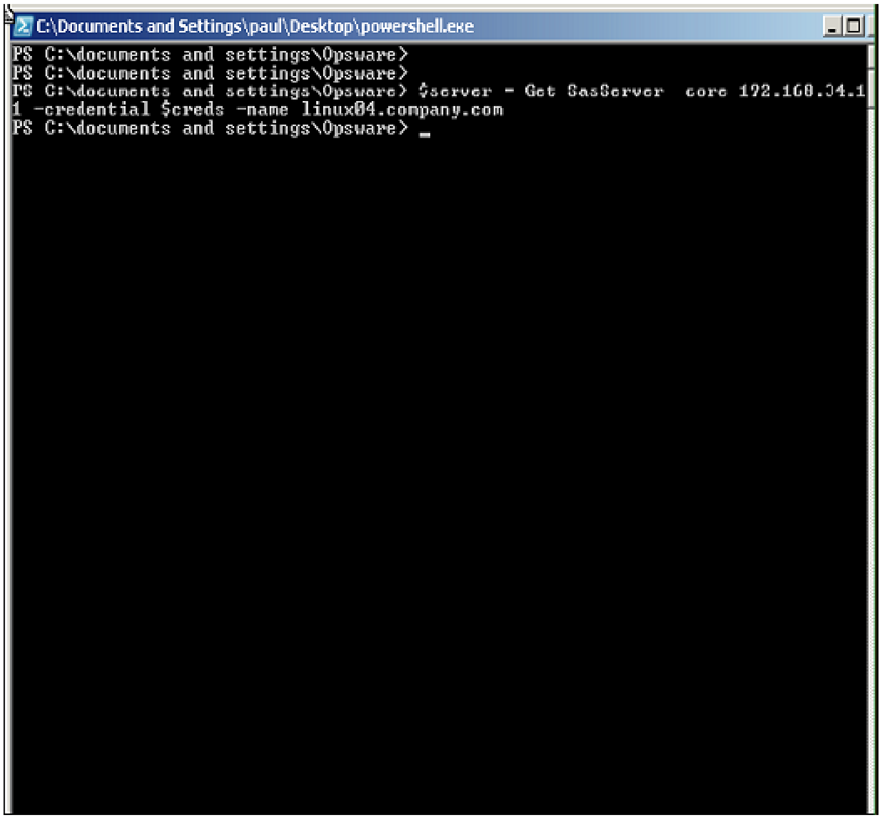
Storing the SA Credentials in a PowerShell variable



```
Windows PowerShell
Copyright (C) 2006 Microsoft Corporation. All rights reserved.

PS C:\documents and settings\Opsware> $creds = get-credential

cmdlet get-credential at command pipeline position 1
Supply values for the following parameters:
Credential
User: student35
Password for user student35: ********

PS C:\documents and settings\Opsware> $creds

UserName                                                        Password
--------                                                        --------
student35                                    System.Security.SecureString


PS C:\documents and settings\Opsware>
```

3. Using the Get-SasServer cmdlet, you can retrieve the SA record representing a server as shown in the following figure.

   Using the Get-SasServer cmdlet



```
PS C:\documents and settings\Opsware>
PS C:\documents and settings\Opsware>
PS C:\documents and settings\Opsware> $server = Get SasServer  core 192.168.34.1
1 -credential $creds -name linux04.company.com
PS C:\documents and settings\Opsware> _
```

The returned object is stored in a shell variable.

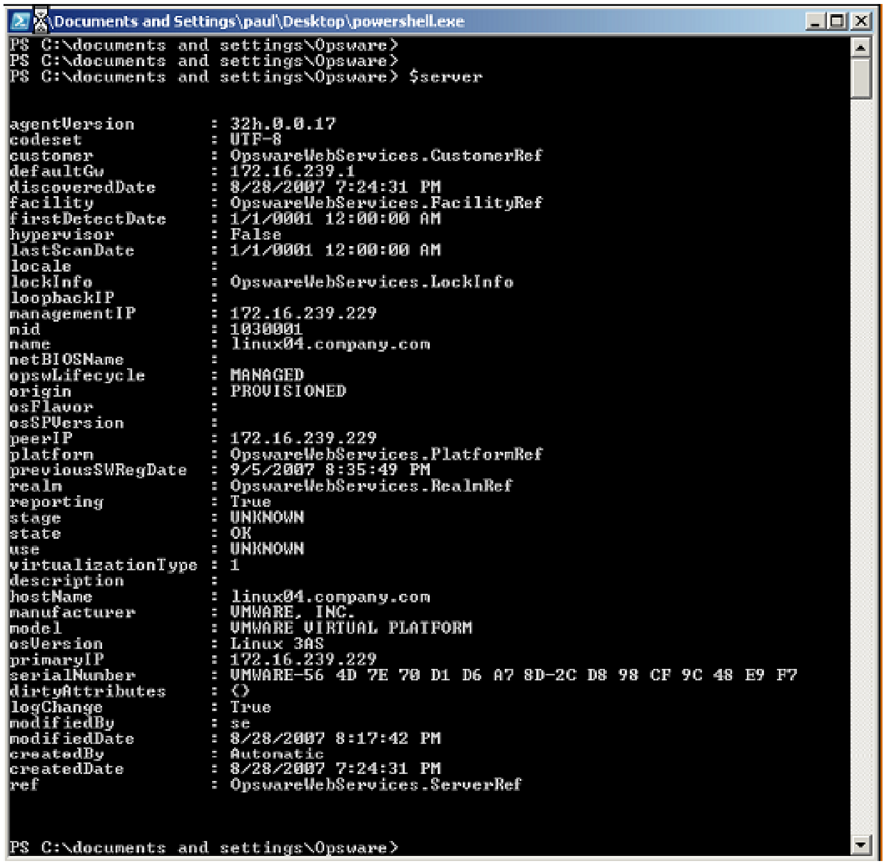
The Get-SasServer cmdlet takes a parameter to identify the Core from which the server data is to be retrieved, a parameter to supply credentials to the core for the operation, identifying and authenticating the user account in whose identity the operation is to be attempted, and a parameter to identify the server being requested.

More information on the Get-SasServer cmdlet arguments or the arguments for any cmdlet can be obtained by using the PowerShell Get-Help base cmdlet, for example:

```
Get-Help Get-SasServer -detailed
```

4. You can now examine the properties of the returned object by entering the name of the shell variable. See the following figure.

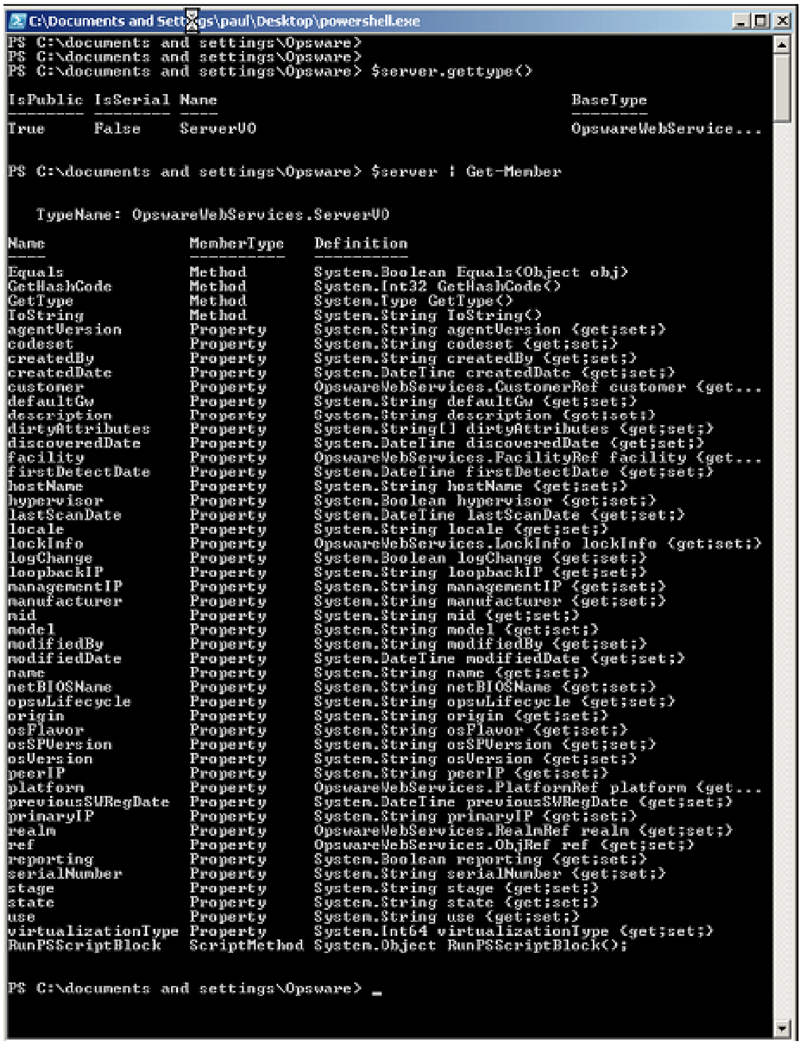
Examining SA Server properties



```
PS C:\documents and settings\Opsware>
PS C:\documents and settings\Opsware>
PS C:\documents and settings\Opsware> $server


agentVersion        : 32h.0.0.17
codeset             : UTF-8
customer            : OpswareWebServices.CustomerRef
defaultGw           : 172.16.239.1
discoveredDate      : 8/28/2007 7:24:31 PM
facility            : OpswareWebServices.FacilityRef
firstDetectDate     : 1/1/0001 12:00:00 AM
hypervisor          : False
lastScanDate        : 1/1/0001 12:00:00 AM
locale              :
lockInfo            : OpswareWebServices.LockInfo
loopbackIP          :
managementIP        : 172.16.239.229
mid                 : 1030001
name                : linux04.company.com
netBIOSName         :
opswLifecycle       : MANAGED
origin              : PROVISIONED
osFlavor            :
osSPVersion         :
peerIP              : 172.16.239.229
platform            : OpswareWebServices.PlatformRef
previousSWRegDate   : 9/5/2007 8:35:49 PM
realm               : OpswareWebServices.RealmRef
reporting           : True
stage               : UNKNOWN
state               : OK
use                 : UNKNOWN
virtualizationType  : 1
description         :
hostName            : linux04.company.com
manufacturer        : VMWARE, INC.
model               : VMWARE VIRTUAL PLATFORM
osVersion           : Linux 3AS
primaryIP           : 172.16.239.229
serialNumber        : VMWARE-56 4D 7E 70 D1 D6 A7 8D-2C D8 98 CF 9C 48 E9 F7
dirtyAttributes     : {}
logChange           : True
modifiedBy          : se
modifiedDate        : 8/28/2007 8:17:42 PM
createdBy           : Automatic
createdDate         : 8/28/2007 7:24:31 PM
ref                 : OpswareWebServices.ServerRef


PS C:\documents and settings\Opsware>
```

5. List the object’s properties, the types of the properties and the methods that can be called on the object from a PowerShell script as shown in following figure.

   Listing an object’s properties

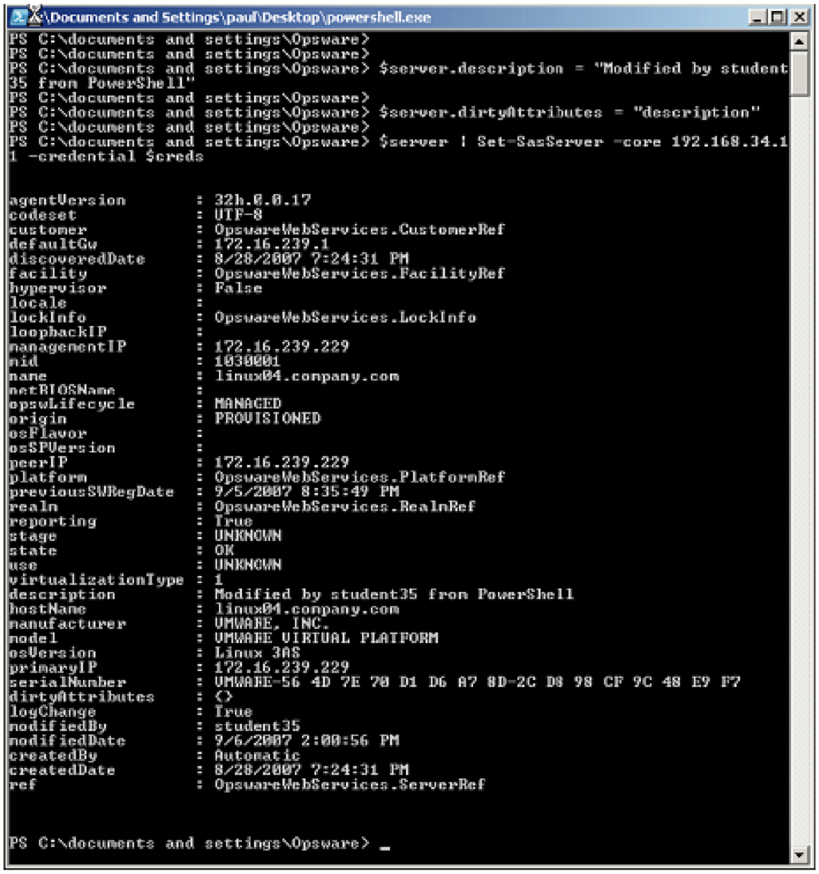


```
PS C:\documents and settings\Opsware>
PS C:\documents and settings\Opsware>
PS C:\documents and settings\Opsware> $server.gettype()

IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     False    ServerVO                                 OpswareWebService...


PS C:\documents and settings\Opsware> $server | Get-Member


   TypeName: OpswareWebServices.ServerVO

Name               MemberType   Definition
----               ----------   ----------
Equals             Method       System.Boolean Equals(Object obj)
GetHashCode        Method       System.Int32 GetHashCode()
GetType            Method       System.Type GetType()
ToString           Method       System.String ToString()
agentVersion       Property     System.String agentVersion {get;set;}
codeset            Property     System.String codeset {get;set;}
createdBy          Property     System.String createdBy {get;set;}
createdDate        Property     System.DateTime createdDate {get;set;}
customer           Property     OpswareWebServices.CustomerRef customer {get...
defaultGw          Property     System.String defaultGw {get;set;}
description        Property     System.String description {get;set;}
dirtyAttributes    Property     System.String[] dirtyAttributes {get;set;}
discoveredDate     Property     System.DateTime discoveredDate {get;set;}
facility           Property     OpswareWebServices.FacilityRef facility {get...
firstDetectDate    Property     System.DateTime firstDetectDate {get;set;}
hostName           Property     System.String hostName {get;set;}
hypervisor         Property     System.Boolean hypervisor {get;set;}
lastScanDate       Property     System.DateTime lastScanDate {get;set;}
locale             Property     System.String locale {get;set;}
lockInfo           Property     OpswareWebServices.LockInfo lockInfo {get;set;}
logChange          Property     System.Boolean logChange {get;set;}
loopbackIP         Property     System.String loopbackIP {get;set;}
managementIP       Property     System.String managementIP {get;set;}
manufacturer       Property     System.String manufacturer {get;set;}
mid                Property     System.String mid {get;set;}
model              Property     System.String model {get;set;}
modifiedBy         Property     System.String modifiedBy {get;set;}
modifiedDate       Property     System.DateTime modifiedDate {get;set;}
name               Property     System.String name {get;set;}
netBIOSName        Property     System.String netBIOSName {get;set;}
opswLifecycle      Property     System.String opswLifecycle {get;set;}
origin             Property     System.String origin {get;set;}
osFlavor           Property     System.String osFlavor {get;set;}
osSPVersion        Property     System.String osSPVersion {get;set;}
osVersion          Property     System.String osVersion {get;set;}
peerIP             Property     System.String peerIP {get;set;}
platform           Property     OpswareWebServices.PlatformRef platform {get...
previousSWRegDate  Property     System.DateTime previousSWRegDate {get;set;}
primaryIP          Property     System.String primaryIP {get;set;}
realm              Property     OpswareWebServices.RealmRef realm {get;set;}
ref                Property     OpswareWebServices.ObjRef ref {get;set;}
reporting          Property     System.Boolean reporting {get;set;}
serialNumber       Property     System.String serialNumber {get;set;}
stage              Property     System.String stage {get;set;}
state              Property     System.String state {get;set;}
use                Property     System.String use {get;set;}
virtualizationType Property     System.Int64 virtualizationType {get;set;}
RunPSScriptBlock   ScriptMethod System.Object RunPSScriptBlock();


PS C:\documents and settings\Opsware> _
```

6. You can modify the object’s **Description** attribute in Windows PowerShell, then call the Set-SasServer cmdlet and pass the modified ServerVO object to the cmdlet. This cmdlet will take the ServerVO object and update the managed server record in the SA Core. The Set-SasServer cmdlet takes parameters that identify the SA Core to which the updated data is to be written and credentials identifying the SA user account under whose identity the operation is executed.

   At the end of the update operation, the updated ServerVO is returned to Windows PowerShell and the properties are displayed at the prompt as shown in the following figure.

   Modifying an object’s description



\Documents and Settings\paul\Desktop\powershell.exe

```
PS C:\documents and settings\Opsware>
PS C:\documents and settings\Opsware>
PS C:\documents and settings\Opsware> $server.description = "Modified by student
35 from PowerShell"
PS C:\documents and settings\Opsware>
PS C:\documents and settings\Opsware> $server.dirtyAttributes = "description"
PS C:\documents and settings\Opsware>
PS C:\documents and settings\Opsware> $server | Set-SasServer -core 192.168.34.1
1 -credential $creds


agentVersion        : 32h.0.0.17
codeset             : UTF-8
customer            : OpswareWebServices.CustomerRef
defaultGw           : 172.16.239.1
discoveredDate      : 8/28/2007 7:24:31 PM
facility            : OpswareWebServices.FacilityRef
hypervisor          : False
locale              :
lockInfo            : OpswareWebServices.LockInfo
loopbackIP          :
managementIP        : 172.16.239.229
mid                 : 1030001
name                : linux04.company.com
netBIOSName         :
opswLifecycle       : MANAGED
origin              : PROVISIONED
osFlavor            :
osSPVersion         :
peerIP              : 172.16.239.229
platform            : OpswareWebServices.PlatformRef
previousSWRegDate   : 9/5/2007 8:35:49 PM
realm               : OpswareWebServices.RealmRef
reporting           : True
stage               : UNKNOWN
state               : OK
use                 : UNKNOWN
virtualizationType  : 1
description         : Modified by student35 from PowerShell
hostName            : linux04.company.com
manufacturer        : VMWARE, INC.
model               : VMWARE VIRTUAL PLATFORM
osVersion           : Linux 3AS
primaryIP           : 172.16.239.229
serialNumber        : VMWARE-56 4D 7E 70 D1 D6 A7 8D-2C D8 98 CF 9C 48 E9 F7
dirtyAttributes     : {}
logChange           : True
modifiedBy          : student35
modifiedDate        : 9/6/2007 2:00:56 PM
createdBy           : Automatic
createdDate         : 8/28/2007 7:24:31 PM
ref                 : OpswareWebServices.ServerRef


PS C:\documents and settings\Opsware> _
```

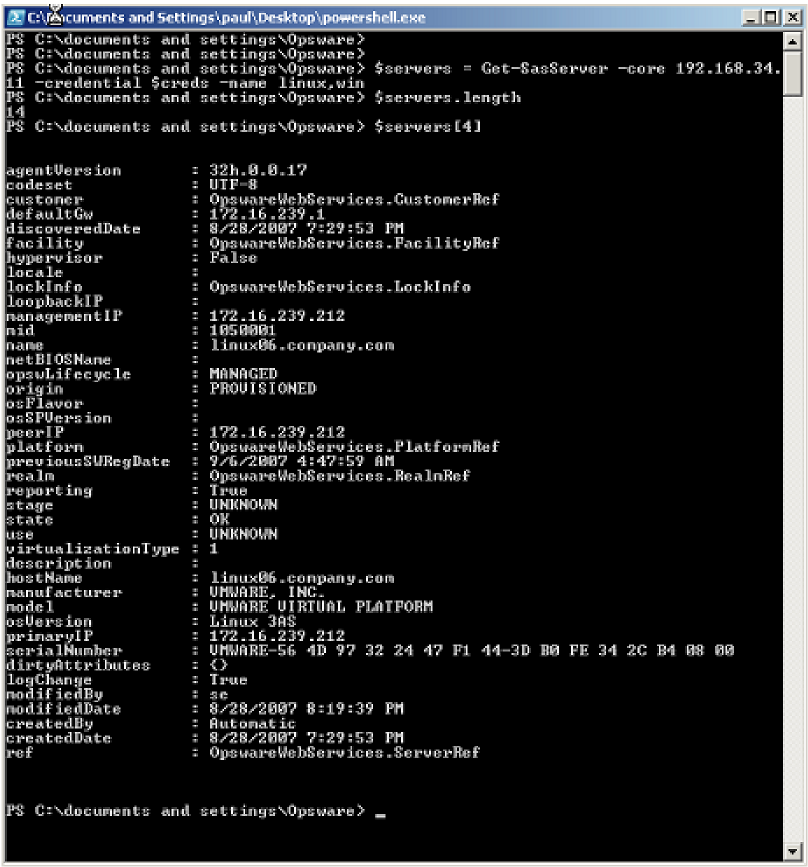
Scenario 2

This scenario demonstrates retrieving all managed server data from the SA Core and displaying it in Microsoft Excel.

1. Use the `Get-SasServer` cmdlet to retrieve ServerVOs for each Linux and Windows managed server from the SA Core. In the session below, the `-name` parameter is used to supply a list of name matching filters, for example, `-name linux`,`win`, to the SA Core.

   The `Get-SasServer` cmdlet returns an array of ServerVOs that is, in this example, 14 items in length. You can index into this array to examine any one of the ServerVO objects. See the following figure.

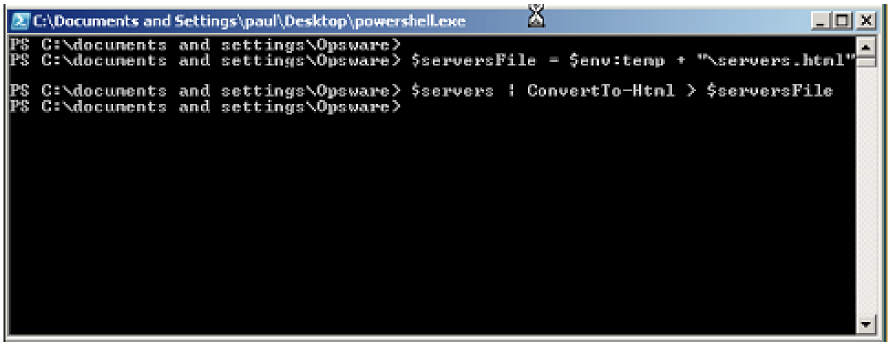
   Using the Get-SasServer cmdlet with a name filter



```
PS C:\documents and settings\Opsware>
PS C:\documents and settings\Opsware>
PS C:\documents and settings\Opsware> $servers = Get-SasServer -core 192.168.34.
11 -credential $creds -name linux,win
PS C:\documents and settings\Opsware> $servers.length
14
PS C:\documents and settings\Opsware> $servers[4]


agentVersion        : 32h.0.0.17
codeset             : UTF-8
customer            : OpswareWebServices.CustomerRef
defaultGw           : 172.16.239.1
discoveredDate      : 8/28/2007 7:29:53 PM
facility            : OpswareWebServices.FacilityRef
hypervisor          : False
locale              :
lockInfo            : OpswareWebServices.LockInfo
loopbackIP          :
managementIP        : 172.16.239.212
mid                 : 1050001
name                : linux06.company.com
netBIOSName         :
opswLifecycle       : MANAGED
origin              : PROVISIONED
osFlavor            :
osSPVersion         :
peerIP              : 172.16.239.212
platform            : OpswareWebServices.PlatformRef
previousSWRegDate   : 9/6/2007 4:47:59 AM
realm               : OpswareWebServices.RealmRef
reporting           : True
stage               : UNKNOWN
state               : OK
use                 : UNKNOWN
virtualizationType  : 1
description         :
hostName            : linux06.company.com
manufacturer        : VMWARE, INC.
model               : VMWARE VIRTUAL PLATFORM
osVersion           : Linux 3AS
primaryIP           : 172.16.239.212
serialNumber        : VMWARE-56 4D 97 32 24 47 F1 44-3D B0 FE 34 2C B4 08 00
dirtyAttributes     : {}
logChange           : True
modifiedBy          : se
modifiedDate        : 8/28/2007 8:19:39 PM
createdBy           : Automatic
createdDate         : 8/28/2007 7:29:53 PM
ref                 : OpswareWebServices.ServerRef


PS C:\documents and settings\Opsware> _
```

2. Now you can format the ServerVO data as HTML and save to a temporary file. The temporary file is created in the TEMP directory. In a PowerShell session, to get the value of the %TEMP% environment variable, enter $env:temp. See the following figure.

   Converting ServerVO Data to HTML and saving to a temporary file

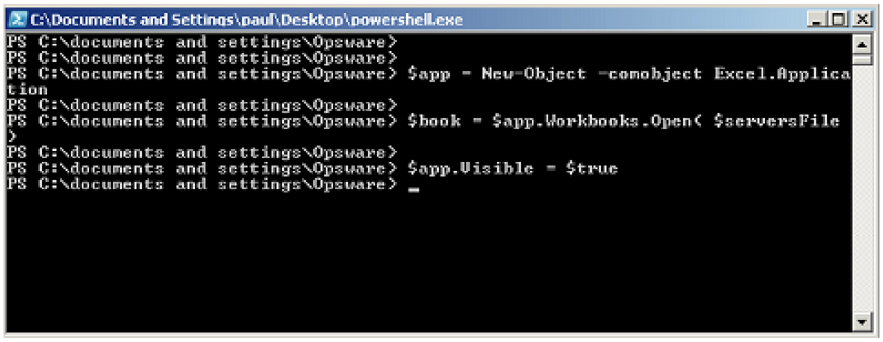


```
PS C:\documents and settings\Opsware>
PS C:\documents and settings\Opsware> $serversFile = $env:temp + "\servers.html"

PS C:\documents and settings\Opsware> $servers | ConvertTo-Html > $serversFile
PS C:\documents and settings\Opsware>
```

3. Using the New-Object base Windows PowerShell cmdlet you can launch Microsoft Excel, then create a new workbook inside this instance of Excel, and populate the workbook from the contents of the temporary file. Finally, set the running Excel instance to be visible. This will cause Excel to

come to the foreground. Now you can sort the data by date, column value, etc., to determine, for example, the date on which each server came under management in the Core. See the following figure.

Using the New-Object cmdlet to launch Microsoft Excel



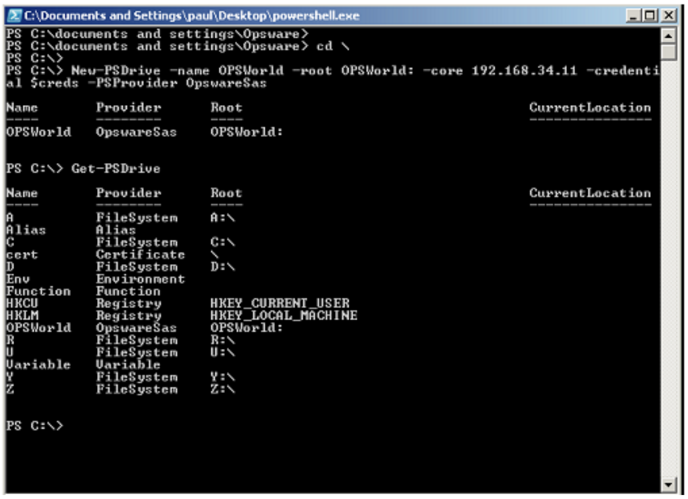
```
PS C:\documents and settings\Opsware>
PS C:\documents and settings\Opsware>
PS C:\documents and settings\Opsware> $app = New-Object -comobject Excel.Applica
tion
PS C:\documents and settings\Opsware>
PS C:\documents and settings\Opsware> $book = $app.Workbooks.Open( $serversFile
)
PS C:\documents and settings\Opsware>
PS C:\documents and settings\Opsware> $app.Visible = $true
PS C:\documents and settings\Opsware> _
```

Scenario 3

This scenario demonstrates mounting the SA Core as a Windows PowerShell PSDrive, navigating to the SA **Jobs** folder and retrieving its contents.

1. Mount the SA Core as a Windows PowerShell PSDrive. PowerShell allows different data stores or repositories to be navigated as if they were a file system. In this scenario, you *mount* the SA Core, specifically the managed environment data store, as if it were a drive named OPSWorld. The windows PowerShell base system then calls the PowerShell SAS Provider, -PSProvider OpswareSas, whenever data is read from or written to this virtual file system — or when the file system is navigated by a client. See the following figure.

   Mounting the SA Core as a Windows PowerShell PSDrive

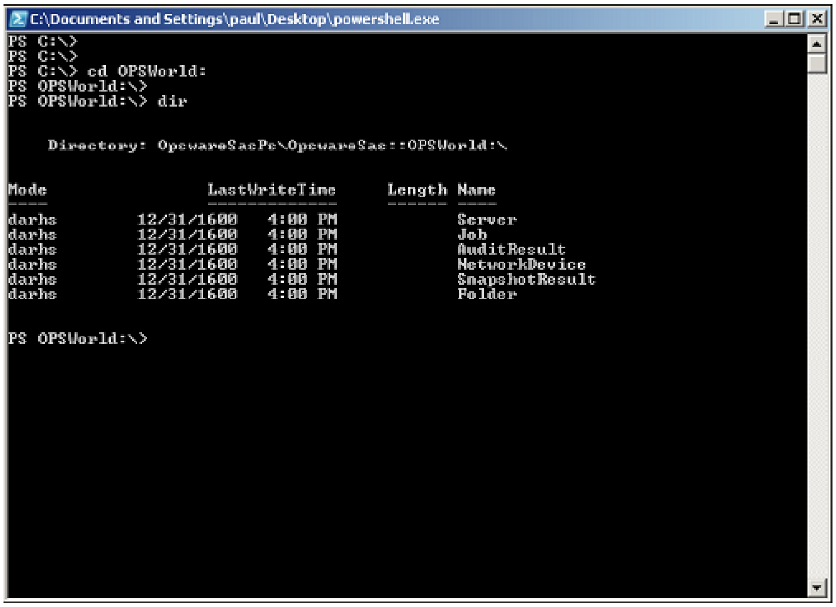


C:\Documents and Settings\paul\Desktop\powershell.exe

```
PS C:\documents and settings\Opsware>
PS C:\documents and settings\Opsware> cd \
PS C:\>
PS C:\> New-PSDrive -name OPSWorld -root OPSWorld: -core 192.168.34.11 -credenti
al $creds -PSProvider OpswareSas

Name       Provider      Root                                   CurrentLocation
----       --------      ----                                   ---------------
OPSWorld   OpswareSas    OPSWorld:


PS C:\> Get-PSDrive

Name       Provider      Root                                   CurrentLocation
----       --------      ----                                   ---------------
A          FileSystem    A:\
Alias      Alias
C          FileSystem    C:\
cert       Certificate   \
D          FileSystem    D:\
Env        Environment
Function   Function
HKCU       Registry      HKEY_CURRENT_USER
HKLM       Registry      HKEY_LOCAL_MACHINE
OPSWorld   OpswareSas    OPSWorld:
R          FileSystem    R:\
U          FileSystem    U:\
Variable   Variable
Y          FileSystem    Y:\
Z          FileSystem    Z:\


PS C:\>
```

2. Change directory to the newly mounted drive and obtain a directory listing. dir is a PowerShell alias for the Get-ChildItem cmdlet. See the following figure.

   DIR as an alias for the Get-Child cmdlet

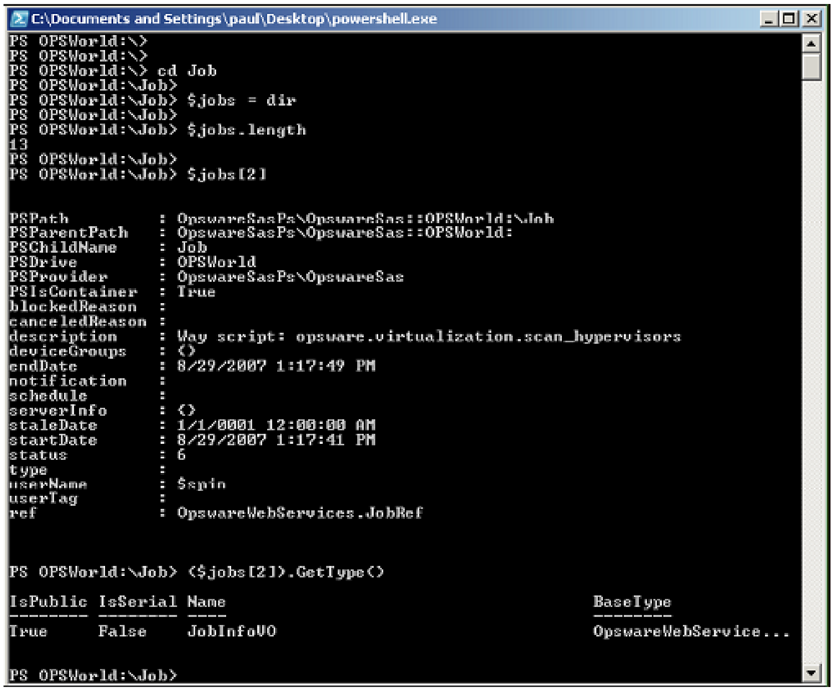
   

   C:\Documents and Settings\paul\Desktop\powershell.exe

   ```
   PS C:\>
   PS C:\>
   PS C:\> cd OPSWorld:
   PS OPSWorld:\>
   PS OPSWorld:\> dir


       Directory: OpswareSasPs\OpswareSas::OPSWorld:\


   Mode                LastWriteTime     Length Name
   ----                -------------     ------ ----
   darhs        12/31/1600   4:00 PM            Server
   darhs        12/31/1600   4:00 PM            Job
   darhs        12/31/1600   4:00 PM            AuditResult
   darhs        12/31/1600   4:00 PM            NetworkDevice
   darhs        12/31/1600   4:00 PM            SnapshotResult
   darhs        12/31/1600   4:00 PM            Folder


   PS OPSWorld:\>
   ```

3. Change directory to the **Jobs** folder, get a directory listing, and save the directory listing as a shell variable. This shell variable will contain an array of JobInfoVO objects from the Core into which you can index.

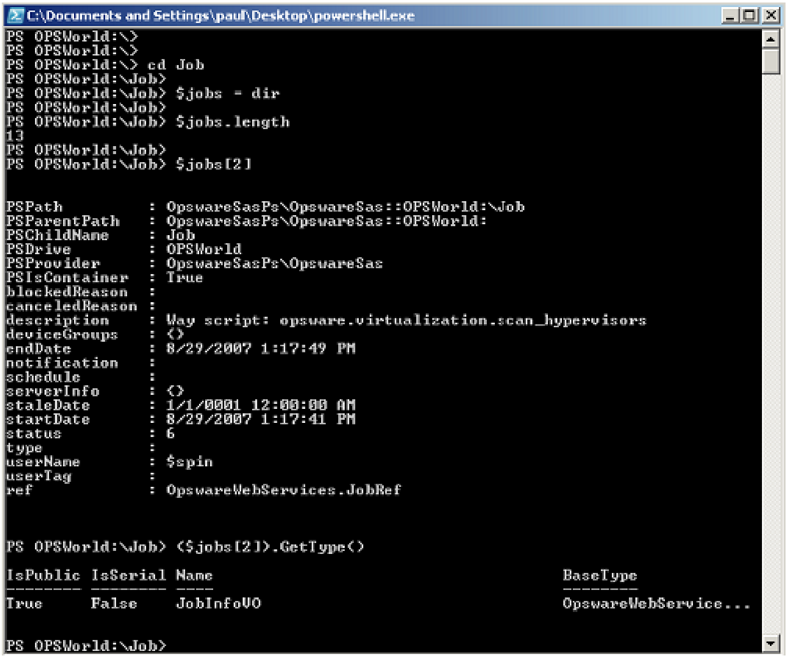
   Save a Directory Listing as a PowerShell variable



```
PS OPSWorld:\>
PS OPSWorld:\>
PS OPSWorld:\> cd Job
PS OPSWorld:\Job>
PS OPSWorld:\Job> $jobs = dir
PS OPSWorld:\Job>
PS OPSWorld:\Job> $jobs.length
13
PS OPSWorld:\Job>
PS OPSWorld:\Job> $jobs[2]


PSPath          : OpswareSasPs\OpswareSas::OPSWorld:\Job
PSParentPath    : OpswareSasPs\OpswareSas::OPSWorld:
PSChildName     : Job
PSDrive         : OPSWorld
PSProvider      : OpswareSasPs\OpswareSas
PSIsContainer   : True
blockedReason   :
canceledReason  :
description     : Way script: opsware.virtualization.scan_hypervisors
deviceGroups    : {}
endDate         : 8/29/2007 1:17:49 PM
notification    :
schedule        :
serverInfo      : {}
staleDate       : 1/1/0001 12:00:00 AM
startDate       : 8/29/2007 1:17:41 PM
status          : 6
type            :
userName        : $spin
userTag         :
ref             : OpswareWebServices.JobRef



PS OPSWorld:\Job> ($jobs[2]).GetType()

IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     False    JobInfoVO                                OpswareWebService...


PS OPSWorld:\Job>
```

4. Change directory to the C: drive and remove the OPSWorld PSDrive.

Removing the OPSWorld PSDrive



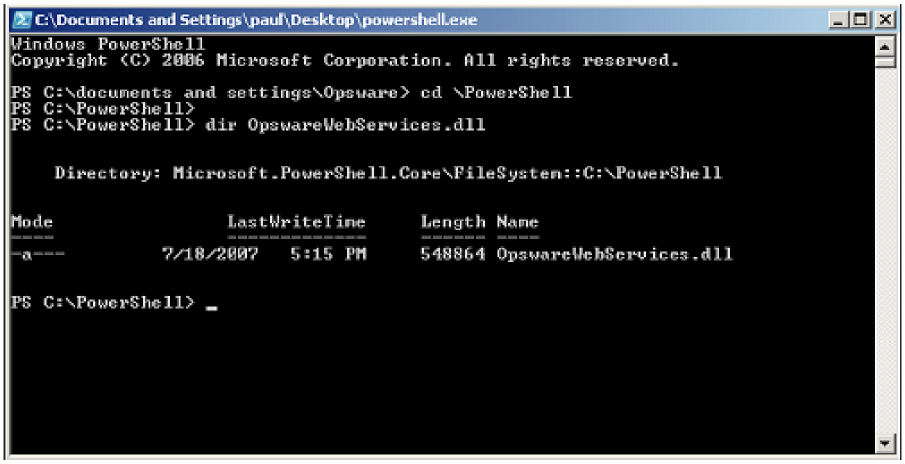
```
PS OPSWorld:\>
PS OPSWorld:\>
PS OPSWorld:\> cd Job
PS OPSWorld:\Job>
PS OPSWorld:\Job> $jobs = dir
PS OPSWorld:\Job>
PS OPSWorld:\Job> $jobs.length
13
PS OPSWorld:\Job>
PS OPSWorld:\Job> $jobs[2]


PSPath          : OpswareSasPs\OpswareSas::OPSWorld:\Job
PSParentPath    : OpswareSasPs\OpswareSas::OPSWorld:
PSChildName     : Job
PSDrive         : OPSWorld
PSProvider      : OpswareSasPs\OpswareSas
PSIsContainer   : True
blockedReason   :
canceledReason  :
description     : Way script: opsware.virtualization.scan_hypervisors
deviceGroups    : {}
endDate         : 8/29/2007 1:17:49 PM
notification    :
schedule        :
serverInfo      : {}
staleDate       : 1/1/0001 12:00:00 AM
startDate       : 8/29/2007 1:17:41 PM
status          : 6
type            :
userName        : $spin
userTag         :
ref             : OpswareWebServices.JobRef



PS OPSWorld:\Job> ($jobs[2]).GetType()

IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     False    JobInfoVO                                OpswareWebService...


PS OPSWorld:\Job>
```

Scenario 4

This scenario describes examining all the types of SA objects available inside the Windows PowerShell environment.

1. Locate the .NET assembly containing the PowerShell SAS Provider and cmdlets. See the following figure.

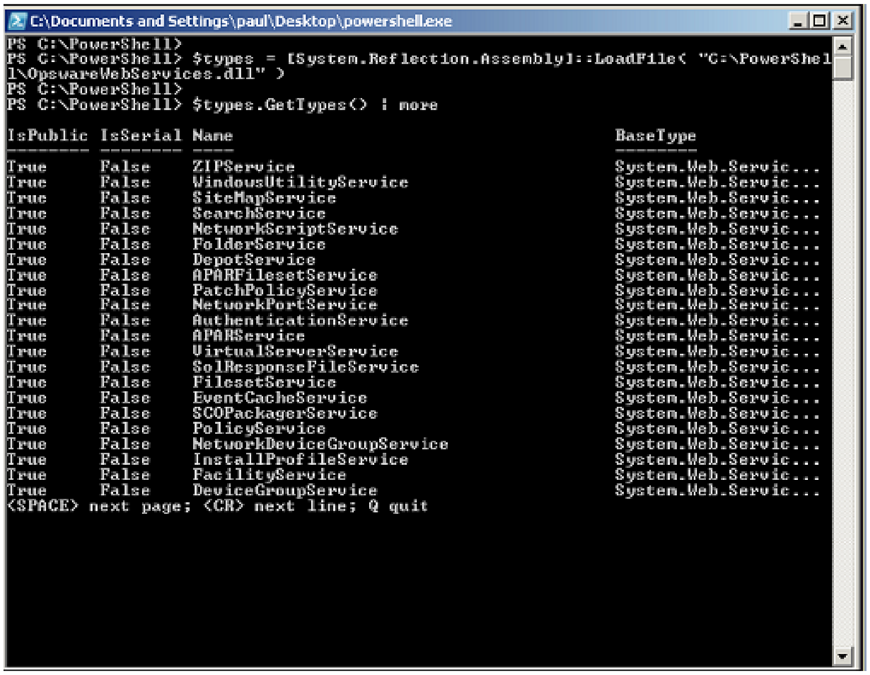
   Locating the .NET Assembly containing the PowerShell SAS Provider and cmdlets

   ```
   Windows PowerShell
   Copyright (C) 2006 Microsoft Corporation. All rights reserved.

   PS C:\documents and settings\Opsware> cd \PowerShell
   PS C:\PowerShell>
   PS C:\PowerShell> dir OpswareWebServices.dll


       Directory: Microsoft.PowerShell.Core\FileSystem::C:\PowerShell


   Mode                LastWriteTime     Length Name
   ----                -------------     ------ ----
   -a---          7/18/2007   5:15 PM     548864 OpswareWebServices.dll


   PS C:\PowerShell> _
   ```

2. Using .NET Reflection, load the .NET assembly and examine the loaded types. This displays all the SA types that are available for use in the Windows PowerShell environment. See the following figure.

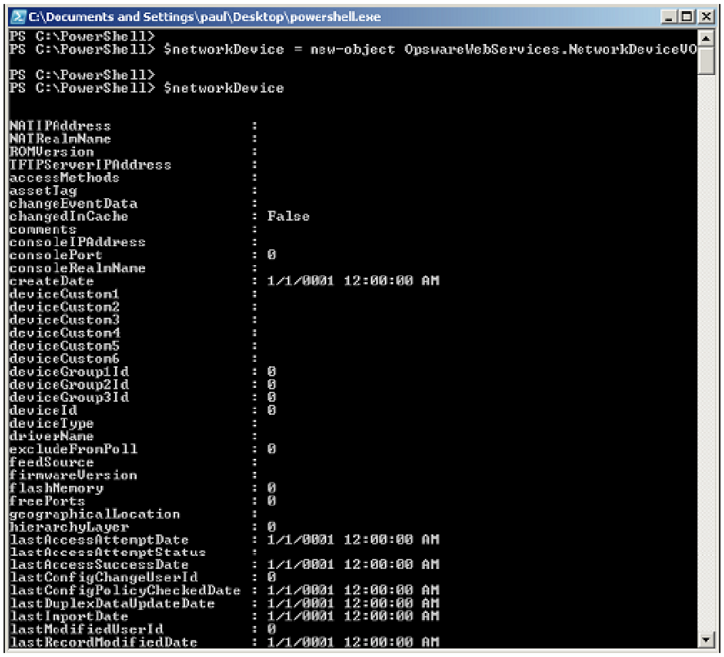
Loading the .NET Assembly and examining the types



```
PS C:\PowerShell>
PS C:\PowerShell> $types = [System.Reflection.Assembly]::LoadFile( "C:\PowerShel
l\OpswareWebServices.dll" )
PS C:\PowerShell>
PS C:\PowerShell> $types.GetTypes() | more

IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     False    ZIPService                               System.Web.Servic...
True     False    WindowsUtilityService                    System.Web.Servic...
True     False    SiteMapService                           System.Web.Servic...
True     False    SearchService                            System.Web.Servic...
True     False    NetworkScriptService                     System.Web.Servic...
True     False    FolderService                            System.Web.Servic...
True     False    DepotService                             System.Web.Servic...
True     False    APARFilesetService                       System.Web.Servic...
True     False    PatchPolicyService                       System.Web.Servic...
True     False    NetworkPortService                       System.Web.Servic...
True     False    AuthenticationService                    System.Web.Servic...
True     False    APARService                              System.Web.Servic...
True     False    VirtualServerService                     System.Web.Servic...
True     False    SolResponseFileService                   System.Web.Servic...
True     False    FilesetService                           System.Web.Servic...
True     False    EventCacheService                        System.Web.Servic...
True     False    SCOPackagerService                       System.Web.Servic...
True     False    PolicyService                            System.Web.Servic...
True     False    NetworkDeviceGroupService                System.Web.Servic...
True     False    InstallProfileService                    System.Web.Servic...
True     False    FacilityService                          System.Web.Servic...
True     False    DeviceGroupService                       System.Web.Servic...
<SPACE> next page; <CR> next line; Q quit
```

3. Create an instance of a NetworkDeviceVO. This is a nascent NetworkDeviceVO, showing all of the attributes of a network device available for scripting, reporting etc. in the PowerShell environment. See the following figure.

Creating an Instance of a NetworkDeviceVO



```
PS C:\PowerShell>
PS C:\PowerShell> $networkDevice = new-object OpswareWebServices.NetworkDeviceVO

PS C:\PowerShell>
PS C:\PowerShell> $networkDevice


NATIPAddress                  :
NATRealmName                  :
ROMVersion                    :
TFTPServerIPAddress           :
accessMethods                 :
assetTag                      :
changeEventData               :
changedInCache                : False
comments                      :
consoleIPAddress              :
consolePort                   : 0
consoleRealmName              :
createDate                    : 1/1/0001 12:00:00 AM
deviceCustom1                 :
deviceCustom2                 :
deviceCustom3                 :
deviceCustom4                 :
deviceCustom5                 :
deviceCustom6                 :
deviceGroup1Id                : 0
deviceGroup2Id                : 0
deviceGroup3Id                : 0
deviceId                      : 0
deviceType                    :
driverName                    :
excludeFromPoll               : 0
feedSource                    :
firmwareVersion               :
flashMemory                   : 0
freePorts                     : 0
geographicalLocation          :
hierarchyLayer                : 0
lastAccessAttemptDate         : 1/1/0001 12:00:00 AM
lastAccessAttemptStatus       :
lastAccessSuccessDate         : 1/1/0001 12:00:00 AM
lastConfigChangeUserId        : 0
lastConfigPolicyCheckedDate   : 1/1/0001 12:00:00 AM
lastDuplexDataUpdateDate      : 1/1/0001 12:00:00 AM
lastImportDate                : 1/1/0001 12:00:00 AM
lastModifiedUserId            : 0
lastRecordModifiedDate        : 1/1/0001 12:00:00 AM
```

Java RMI clients

A Java Remote Invocation (RMI) client can call the methods of the SA API from a server that has network access to the SA core. The server running the client does not have to be an SA core or managed server. When it connects to the core, the client specifies an SA user name and password, much like an end user logging on with the SA Client. The group that the user belongs to determines which SA resources and tasks are available to the client.

This topic is intended for software developers who are familiar with SA fundamentals and the Java programming language.

Setup for Java RMI clients

Before developing Java RMI clients for the SA API, perform the following steps:

1. Install an SA core in a development environment. Do not use a production core.
2. Obtain a development server where you will build and run the Java RMI client.
3. On the development server, install the Java SE 7 SDK.
4. Verify that the development server has a network connection to the SA core server that runs the OCC component.
5. Download the `opswclient.jar` file from the SA core server to your development server. The `opswclient.jar` file contains the Java RMI stubs for the SA API. You include the `opswclient.jar` in the `classpath` option when compiling and running Java RMI clients.
6. To download `opswclient.jar` do one of the following:
   a. Specify the following URL, where *occ_host* is the core server running the OCC component:

      ```
      https://occ_host/twister/opswclient.jar
      ```

   b. Go to the following directory: `/opt/opsware/twist/extlib/client`.

You also need the `spinclient-latest.jar` and the `opsware_common-latest.jar` files. These files can be obtained from a running SA Core in:

`/opt/opsware/twist/lib/`

You must also add these `.jar` files to the classpath parameter when compiling and running these examples.

Sample Java RMI

This section describes a simple Java RMI client named GetServerInfo.

The GetServerInfo client searches for managed servers by full or partial host name, which you specify as a command-line argument. For each managed server found, the client prints out the server's name, management IP address, and OS version.

The GetServerInfo client performs the following steps:

1. Connects to SA:
   ```
   // Set the JNDI provider
   client.setJNDIProvider( "https" , host , ( short ) 1032 , null, newString[] {
   OPSWARE_CA_CRT_PATH } , null ) ;
   // Force a reconnection
   client.getContext( true ) ;
   // Set and authenticate the user
   client.setAPIUser( new APIUserImpl(username , password)) ;
   ```
2. Gets a reference to the ServerService interface:
   ```
   serverSvc = (ServerService)OpswareClient.getService
   (ServerService.class);
   ```
3. Invokes methods on ServerService:
   ```
   ServerRef[] serverRefs = serverSvc.findServerRefs(filter);
   . . .
   ServerVO[] serverVOs = serverSvc.getServerVOs(serverRefs);
   . . .
   System.out.println(serverVOs[i].getName());
   ```

Compiling and running the GetServerInfo example

Before compiling and running the example, perform the following tasks:

1. Obtain the opsware_common-latest.jar, spinclient-latest.jar andopswclient.jar files, as described in Setup for Java RMI Clients.
2. Download the ZIP file that contains the demo program GetServerInfo.java file.
3. To compile the client, specify the opsware_common-latest.jar, spinclient-latest.jar and opswclient.jar files for the classpath parameter:

```
javac -classpath :path/opswclient.jar:path/opsware_common-
latest.jar:path/spinclient-latest.jar GetServerInfo.java
```

4. To run the client, enter the following command, where *target* is the full or partial name of a server managed by SA. Note: the Java classpath separator for windows is "`;`".

   ```
   java -classpath .:path/opswclient.jar:path/opsware_common-
   latest.jar:path/spinclient-latest.jar \GetServerInfo [options]    target
   ```

   In the following example, `GetServerInfo` connects to SA on host `c44` (where the OCC core component runs) and port 443. The program displays information for managed servers with hostnames that contain the string `opsw`.

   ```
   java -classpath .:/home/jdoe/opswclient.jar:/home/jdoe/opsware_common-
   latest.jar:/home/jdoe/spinclient-latest.jar \GetServerInfo --host
   c44.dev.example.com --port 443 opsw
   ```

5. Respond to the prompts for the SA user name and password. The SA user must have read permissions for the servers that match the *target* specified on the command line.

Web Services clients

Overview of Web Services clients

The SA API supports Web Services, a programming environment built on open industry standards such as SOAP (Simple Object Access Protocol) and WSDL (Web Services Definition Language). You can create Web Services clients in a variety of programming languages such as Perl and C# (as shown later in this section) or with Web Services-enabled development environments such as Microsoft Visual Studio .NET.

This topic is intended for software developers who are familiar with SA fundamentals and Web Services development.

Programming language bindings provided in this release

This release of SA includes Web Services client stubs for C#. Web Services clients written in Perl do not require client stubs.

This release does not include Web Services client stubs for Java or Python. However, Java clients can access the SA API through RMI and Python clients through Pytwist, as described in the preceding sections.

URLs for service locations and WSDLs

Clients access the Web Services at URLs with the following syntax, where host is the server running the OCC core component and *port* is for the HTTPS proxy. (The default proxy port is 443). The *packageName* corresponds to the Java library that the service belongs to.

```
https://host:port/osapi/packageName/WebServiceName
```

The WSDL files are at URLs with the following syntax:

```
https://host:port/osapi/packageName/WebServiceName?WSDL
```

For example, the following URLs point to the FolderService location and WSDL:

```
https://occ.c38.example.com:443/osapi/com/opsware/folder/FolderService
```

```
https://occ.c39.example.com:443/osapi/com/opsware/folder/FolderService?wsdl
```

The SOAP binding style is RPC (Remote Procedure Call) and the transport protocol is HTTPS.

Security for Web Services clients

Like other clients of the SA API, Web Services clients must be authenticated and authorized to perform operations in SA. Communication between clients and the Web Services component in the SA core is encrypted. Access is restricted to HTTPS clients through the HTTPS proxy port of the OCC core component. (The default port is 443.)

Overloaded operations

The SA API has overloaded operations, but the WSDL 2.0 specifications do not support overloading. An overloaded operation in the SA API is exposed by the Web Service as a single operation.

Java interface support

The SA API uses Java interfaces, but Web Services does not support interfaces. As a workaround, the WSDL files map interfaces to `xsd:anyType`. For clients coded in object-oriented programming languages such as C#, if an API method returns an interface, the return type must be cast to a concrete class. Arrays of interfaces are converted to `Object[]`; specific types of the array members are preserved through serialization/deserialization. For a C# code example, see "Handle interface return types" on page 182.

Unsupported data types

The following data types are used by the SA API but are not supported by SOAP:

```
java.util.Properties
com.opsware.common.ModifiableMap
com.opsware.acm.ValueSet
com.opsware.swmgmt.PolicyOverrideFilter
```

Methods omitted from Web Services

The following SA API methods use unsupported data types as parameters or return types. As a result, they are not exposed as operations in the Web Services.

```
com.opsware.custattr.CustomAttribute.getCustAttrs
com.opsware.custattr.CustomAttribute.setCustAttrs
com.opsware.custattr.CustomField.getCustomFields
com.opsware.custattr.CustomField.setCustomFields
com.opsware.pkg.Patch.getPolicyOverrideRefs
```

Partial support for java.util.Map

Axis converts `java.util.Ma`p to `apachesoap:Map`, which is a collection of key-value pairs. With .NET, this conversion does not work. C# clients, for example, will receive an empty array of key-value pairs. However, this conversion does work with Soap::Lite in Perl. Therefore, SA API methods that use `java.util.Ma`p are available as operations in the Web Services.

The following methods use `java.util.Ma`p as parameters or return types:

```
com.opsware.acm.GroupConfigurable.getApplicationInstances
com.opsware.acm.ServerConfigurable.getCustAttrsWithRC
com.opsware.compliance.sco.CMLSnapshot.getValueSet
com.opsware.compliance.sco.CMLSnapshot.setValueSet
com.opsware.compliance.sco.SnapshotResultService.remediateCMLSnapshot
com.opsware.custattr.VirtualColumnVO.getConfigInfo
com.opsware.custattr.VirtualColumnVO.setConfigInfo
```

Methods in VOs with unsupported data types

The following methods of VOs use unsupported data types as parameters or return types:

```
com.opsware.acm.ApplicationInstanceVO.getValueset
com.opsware.acm.ApplicationInstanceVO.setValueset
com.opsware.acm.ConfigurableVO.getValueset
com.opsware.acm.ConfigurableVO.setValueset
com.opsware.virtualization.VirtualConfigNode.getProperties
com.opsware.virtualization.VirtualConfigNode.setProperties
com.opsware.virtualization.VirtualServerConfig.getProperties
com.opsware.virtualization.VirtualServerConfig.setProperties
```

Invoke setDirtyAtrributes when creating or updating VOs

Web Services clients must invoke `setDirtyAttributes` before invoking a `create` or `update` method on a service. The `setDirtyAttributes` method explicitly the marks the attributes (fields) of a VO that need to be set by the `create` or `update` invocation. The attribute names specified by `setDirtyAttributes` are case sensitive.

For example, to modify the `description` attribute of a `FolderVO` object, the following code invokes `setDirtyAttributes` before it invokes `update`:

```
// fs is FolderService
FolderVO folderVO = fs.getFolderVO(folderRef);
folderVO.setDescription("credit card processing");
folderVO.setDirtyAttributes(new String[]{"description"});
fs.update(folderRef, folderVO, true, true);
```

Invoking `setDirtyAttributes` is required for Web Services clients because of the way Axis deserializes XML objects from XML. If `setDirtyAttributes` is not invoked, Axis calls setters on all attributes of the VO, including read-only attributes, resulting in a `ReadOnlyException`.

Compatibility with SA Web Services API 2.2

The SA Web Services API 2.2 is not compatible with the SA API described in this guide. The method signatures, services, WSDLs, and port bindings are not the same. If you are creating new Web Services clients, be sure to use the SA API, not the SA Web Services API 2.2.

Perl Web Services clients

This section contains step-by-step instructions and sample code for creating Perl Web Services clients that access the SA API.

Required software for Perl clients

Your development environment must have the following Perl modules:

- Crypt-SSLeay-0.57
- IO-Socket-SSL-1.31
- Net-SSLeay-1.35
- HTML-Parser-3.64
- MIME-Base64-3.08
- URI-1.40
- libwww-perl-5.833
- SOAP-Lite-0.710

Depending on your Perl version, newer versions of these modules could be required.

> **Caution:** If the `"500 SSL negotiation failed:"` error persists, then OpenSSL needs to be updated to version 1.0.1 or higher. For the RHEL family, OpenSSL needs to be updated to version 1.0.1e-30 or higher.

Running the Perl demo program

To run the demo program, perform the following steps:

1. From the support site, obtain the **SA_Platform_Developer_Guide_examples.zip** file bundled with the **All Manuals Download SA 10.5** folder.
2. Obtain the demo program `uapisample.pl` file from `SA_Platform_Developer_Guide_examples.zip\SA_Platform_Developer_Guide_examples\api_examples\web_services\perl`.
3. Edit the `uapisample.pl` file, changing the hard-coded values for `host`, `username`, `password`, and object IDs such as `serverID`.
4. Run `uapisample.pl`.
5. If you receive a "`Certificate Verify Failed`" error, you should uncomment the following line from the sample file and provide a valid path to the certificate file:

   ```
   #$ENV{HTTPS_CA_FILE} = "path_to/opsware-ca.crt";
   ```

   You can find the certificate file from an SA Core in:

   ```
   /var/opt/opsware/crypto/twist/opsware-ca.crt
   ```

Sample Perl code

The following code snippets are from `uapisample.pl`, a Perl program contained in the ZIP file you downloaded previously.

Set up the Service URI

```
# Construct the URI for the service.
#
my $username = "integration";
my $password = "integration";
my $protocol = "https";
my $host = "occ.c38.dev.example.com";
my $port = "443";
my $contextUri = "osapi/com/opsware/";
my $folderServiceName = "folder/FolderService";
my $folderUri = "http://www.example.com/" . $contextUri .
$folderServiceName;
# Create a proxy to the FolderService.
#
my $folderProxy = $protocol . "://" . $username . ":" . $password . "@" .
$host . ":" . $port . "/" . $contextUri . $folderServiceName;
```

Initiate a new service

```
my $folderPort = SOAP::Lite
-> uri($folderUri)
-> proxy($folderProxy);
```

Invoke a service method

```
my $root = $folderPort->getRoot()->result();
print 'Got root folder: ' . $root->{'name'} . "\n";
# Alternative:
my $root = $folderPort->SOAP::getRoot();
print 'Got root folder: ' . $root->{'name'} . "\n";
```

Get a VO

```
$rootVO = $folderPort->getFolderVO(SOAP::Data->name('self')
->value(\SOAP::Data->name('id')->type('long')->value(0)))
->result();
# The preceding call to getFolderVO does not pass a FolderRef
# parameter. If a method such as FolderService.remove accepts a
# FolderRef parameter, use the following code:
#
my $folderToBeRemoved = SOAP::Data->name('self')
```

```
->attr({ 'xmlns:ns_fs' => 'http://folder.example.com/FolderService'}) -
>type('ns_fs:FolderRef')->value(\SOAP::Data->name('id')->type('long') -
>value(123456));
$folderPort->remove($folderToBeRemoved);
# To see the Perl representation of the returned VO, you can use
# the Dumper method. This will help you understand how to
# construct the dirty attributes of a VO for a create or update
# method.
#
use Data::Dumper;
print Dumper($folderVO);
```

Get an array

```
# Construct $folder, the FolderRef before getting the array.
#
my $folder = SOAP::Data->name('self') ->attr({ 'xmlns:ns_fs' => 'http://
folder.example.com'}) ->type('ns_fs:FolderRef')->value(\SOAP::Data-
>name('id')->type('long') ->value($root->{'id'}));
# The getChildren method returns an array of FNodeReference
# objects.
#
my $children = $folderPort->getChildren($folder, SOAP::Data->name('type')-
>type('string')->value(''))->result();
foreach $child (@{$children}){
print 'Get child: ' . $child->{'name'} . "\n";
}
```

Construct an object array

```
# For a function that takes an object array as a parameter,
# such the getVOs method, take the following approach:
# First, construct the Array object elements individually
# and put them in an array.
#
my @refs = [];
foreach my $ref (@{$myRefs}){
       # Assume myRefs was returned from a previous
               # Web Services call.
       my $object = SOAP::Data->name('FacilityRef')
               ->value(\SOAP::Data->name('id')
                       ->type('long')
                       ->value($ref->{'id'}
               )
       )->attr({ 'xmlns:facility' => 'http://locality.example.com'})
       ->type('facility:FacilityRef');
   push @refs, $object;
}
# Second, construct an Array Object and put the array in it.
#
```

```
my $selves = SOAP::Data->name("selves" =>\SOAP::Data->name("element" => @refs)-
>type("facility:FacilityRef"))
               ->attr({ 'xmlns:facility' => 'http://locality.example.com'})
               ->type("facility:ArrayOfFacilityRef");
```

Update or create a VO

```
# This example updates the description attribute of a ServerVO.
#
my $serverID = 40038;
my $server = SOAP::Data->name('self')->value(\SOAP::Data->name('id')-
>type('long')->value($serverID));
# Don’t forget to set dirtyAttributes for the attributes
# you want to update. You also need dirtyAttributes for
# create methods that pass a VO.
#
my @dirtyAttrs = ('description');
my $serverVO = SOAP::Data->name('vo') ->attr({ 'xmlns:ns_ss' => 'http://
server.example.com'}) ->value(\SOAP::Data->value( SOAP::Data-
>name('description')->value('PERL_UPDATE_DESC')->type('string'), SOAP::Data-
>name('logChange')->value('false')->type('boolean'), SOAP::Data-
>name('dirtyAttributes' => \SOAP::Data->name("element" => @dirtyAttrs)-
>type("string")) ->type("ns_ss:ArrayOf_soapenc_string"), ));
my $force = SOAP::Data->name('force')->value('true')->type('boolean');
my $refetch = SOAP::Data->name('refetch')->value('true')->type('boolean');
# Call the update method.
#
print 'Invoking method serverWSPort.update...', "\n";
my $updatedServerVO = $serverWSPort->update(
               $server,
               $serverVO,
               $force,
               $refetch)->result();
print "New description: ", $updatedServerVO->{'description'}, "\n";
```

Handle SOAP faults

```
# Make sure that you turn off on_fault subroutine in the
# "use SOAP::Lite ..." statement.
#
# The fault member of a SOAP return will be set if the Web
# Service call throws an exception.
# The following code tries to get a folder that does not exist:
#
my $testVO = $folderPort->getFolderVO(SOAP::Data->name('self') -
>value(\SOAP::Data->name('id')->type('long')->value(123456)));
if($testVO->fault){
       print $testVO->faultstring . "\n";
       # This will print the error msg.
       print "ExceptionName: " . getExceptionName($testVO) . "\n"; # A
NotFoundException should be displayed here
       # The code that deals with the error goes here....
}
. . .
# The following subroutine extracts the exception name from the
# returned faultdetail.
#
sub getExceptionName {
       my $fault = shift; #get the fault object
       if($fault->faultdetail->{'fault'}){
               return ref($fault->faultdetail->{'fault'});
       }
}
. . .
# As shown in the preceding code, it’s easier to handle SOAP
# faults if you execute functions like this:
#
# my $data = $port->function(...);
# Not like this:
# $port->SOAP::function(...);
# $port->function(...)->result;
```

Construction of Perl objects for Web Services

Before calling a Web Services operation, a Perl client must set up the data structures that are required for the input parameters. The information you need for setting up the data structures is in the API documentation (javadocs) and the service’s WSDL file. The Perl code example in this section shows how to construct the input parameter for the `getServerVO` operation. The step-by-step instructions after the code show where to get the information about the input parameter from the API documentation and the WSDL file.

Source code for calling getServerVO

The following Perl code sets up the input parameter `self` and then calls the `getServerVO` operation. This call retrieves the VO (value object) for the managed server of ID 12345.

```
# Create a top-level SOAP::Data object
#
$self = SOAP::Data->name(’self’)
# The namespace corresponds to the schema of the data type
# of the SOAP:Data object. The name chosen (ns_ss) is
# arbitrary.
#
$self->attr({’xmlns:ns_ss =>
’http://server.example.com/ServerService’});
```

```
# Specify the type (ServerRef) for the parameter self, using the
# name of the namespace from the preceding statement.

#
$self->type(’ns_ss:ServerRef’);
# Create the value for the parameter. The value is a pointer
# to a SOAP::Data object. The number 12345 is the SA ID of a managed server.
#
my $id = SOAP::Data->name(’id’)->type(’long’)->value(12345);
# From the self object, point to the value.
#
$self->value(\$id);
# Finally, call getServerVO:
#
my $data = $serverPort->getServerVO($self);
if($data->fault){
       # Handle exceptions here ...
}
else{
       my $serverVO = $data->result;
}
. . .
```

Location of information for getServerVO setup

To get the information needed to write the code for the call to `getServerVO`, perform the following steps:

1. In a browser, go to the API documentation (javadocs) at the following URL:

   `https://occ_host:1032/twister/docs/index.html`

   The *occ_host* is the IP address or host name of the core server running the Command Center component. (For instructions on invoking methods with the Twister, see "API Documentation and the Twister" on page 37.)

2. Examine the API documentation to determine the input parameters and return value of the method.

   The `getServerVO` method is defined in the interface `com.opsware.server.ServerService`. In the following method signature, note that `getServerVO` accepts a `ServerRef` as a parameter and returns a `ServerVO`:

   ```
   public ServerVO getServerVO(ServerRef self)
          throws java.rmi.RemoteException,
                 NotFoundException,
                 AuthorizationException
   ```

3. In a browser, specify the following URL to open the WSDL file for the `ServerService`:

   `https://occ_host/osapi/com/opsware/server/ServerService?wsdl`

4. In the WSDL file, locate the namespace for the `ServerService`:

   ```
   <schema targetNamespace="http://server.example.com"
   xmlns="http://www.w3.org/2001/XMLSchema">
   ```

   The following Perl statement (from the code listed previously) specifies the namespace:

   ```
   $self->attr({’xmlns:ns_ss =>
   ’http://server.example.com/ServerService’});
   ```

5. In the WSDL file, locate the `getServerVO` operation and note the input message name `getServerVORequest`.

   ```
   <wsdl:operation name="getServerVO" parameterOrder="self">
          <wsdl:input message="impl:getServerVORequest" name="getServerVORequest"/>
          <wsdl:output message="impl:getServerVOResponse" name="getServerVOResponse"/
   >
          <wsdl:fault message="impl:NotFoundException" name="NotFoundException"/>
          <wsdl:fault message="impl:AuthorizationException"
   name="AuthorizationException"/>
   </wsdl:operation>
   ```

6. In the WSDL file, locate the `getServerVORequest` message:

   ```
   <wsdl:message name="getServerVORequest">
          <wsdl:part name="self" type="impl:ServerRef"/>
   </wsdl:message>
   ```

   The `getServerVORequest` message element defines the name (`self`) and type (`ServerRef`) of the input parameter of `getServerVO`. The following Perl statement specifies `ServerRef`:

   ```
   $self->type(’ns_ss:ServerRef’);
   ```

7. In the WSDL file, locate the `complexType` for `ServerRef`:

   ```
   <complexType name="ServerRef">
          <complexContent>
                  <extension base="tns1:ObjRef">
                          <sequence>
                                  <element name="secureResourceTypeName" nillable="true"
   type="soapenc:string"/>
                                  </sequence>
                          </extension>
                  </complexContent>
          </complexType>
   ```

Note that `ServerRef` extends `ObjRef`.

8. In the WSDL file, locate the `complexType` for `ObjRef`:

```
<complexType abstract="true" name="ObjRef">
       <sequence>
               <element name="id" type="xsd:long"/>
               <element name="idAsLong" nillable="true" type="soapenc:long"/>
               <element name="name" nillable="true" type="soapenc:string"/>
       </sequence>
</complexType>
```

In `ObjRef`, note the name (`id`) and type (`long`). These data types are specified in the following Perl statement:

```
my $id = SOAP::Data->name(’id’)->type(’long’)->value(12345);
```

C# Web Services clients

This section contains step-by-step instructions and sample code for creating C# Web Services clients that access the SA API.

Required software for C# clients

To develop C# Web Services clients, your development environment must have the following software:

- Microsoft .NET Framework SDK version 1.1
- C# client stubs for SA API

Obtaining the C# client stubs

SA provides a stub file for each service, for example, `FolderService.cs.` All stubs have the same namespace: `OpswareWebServices`. In addition to the stubs, SA provides `shared.cs`, the file that contains shared classes such as `ServerRef`.

To obtain a ZIP file containing the C# stubs, specify the following URL, where *occ_host* is the core server running the OCC component:

```
https://occ_host:1032/twister/opswcsharpclient.zip
```

The constants defined in services and objects are not defined in the C# stubs. To get information about the constants, use the API documentation (javadocs), as described in "Constant field values" on page 37.

Building the C# demo program

To build the demo program:

1. From the support site, obtain the **SA_Platform_Developer_Guide_examples.zip** file bundled with the **All Manuals Download SA 10.5** folder.

   The **SA_Platform_Developer_Guide_examples.zip** contains the following demo program files at the `SA_Platform_Developer_Guide_examples\api_examples\web_services\csharp` location:

   - `App.config` - Application settings
   - `WebServicesDemo.cs` - Client code that invokes service methods
   - `MyCertificateValidation.cs` - Certificate validation class
2. Create the following directory:
   C:\wsapi
3. From the Visual Studio 2008 Start Page, select New Project and create a project with the following values:
   - Project Type: Visual C# Projects
   - Template: Console Application
   - Name: WSAPIDemo
   - Location: `C:\wsapi`

   This action creates the new directory `C:\wsapi\WSAPIDemo`, which contains some files.
4. In the new project, delete the default `program` and `AssemblyInfo.cs` from the list of objects.
5. Copy the files you obtained in step 1 into the `C:\wsapi\WSAPIDemo` directory.
6. Download the client stubs from the URL specified in "Obtaining the C# client stubs" on the previous page.
7. Copy the C# client stubs into the `C:\wsapi\WSAPIDemo` directory.

8. Add the files copied in the preceding two steps to the WSAPIDemo project:
   - In Visual Studio, from the Project menu, select Add Existing Item.
   - Browse to the directory `C:\wsapi\WSAPIDemo`, and select all the demo files (`.cs` and `.config`).
9. Add a reference to `System.Web.Services.dll`:
   - In Visual Studio, from the Project menu, select Add Reference.
   - Under the .NET tag, browse to Component with Name: `System.Web.Services.dll`.
   - Click `System.Web.Services.dll`, click Select, and then click OK.
10. If you used a different template when creating the project, you might need to add references to `System`, `System.XML`, and `System.Data`. Check the Project References to determine if you need to add these references.
11. In the `App.config` file, change the values for `username`, `password`, `host`, and the hardcoded object IDs such as `serverID`.
12. In Visual Studio, from the Build menu, select Build WSAPIDemo.

> **Caution:** If the target core is running a minimum protocol version of TLSv1.x, the Powershell version (the bound underlying .NET Framework version) must support it. For more information see, https://msdn.microsoft.com/en-us/library/system.net.securityprotocoltype(v=vs.110).aspx
>
> Also, it must be explicitly enabled from the C# application.
>
> Code sample: `System.Net.ServicePointManager.SecurityProtocol |= System.Net.SecurityProtocolType.Tls12 | System.Net.SecurityProtocolType.Tls11;"`

Running the C# demo program

To run the demo program:

1. Open the Visual Studio 2008 command prompt:

   Start > All Programs > Microsoft Visual Studio 2008 >
   Visual Studio Tools > Visual Studio 2008 Command Prompt
2. Change the directory to:

   ```
   C:\wsapi\WSAPIDemo\bin\Debug
   ```

3. Enter the following command:

   ```
   WSAPIDemo.exe
   ```

Sample C# code

The following code snippets are from `WebServicesDemo.cs`, a C# program contained in the Zip file you downloaded previously.

Set up certificate handling

```
# This setup is required just once for the client.
#
ServicePointManager.CertificatePolicy = new MyCertificateValidation();
```

Assign the URL prefix

```
# This is the URL prefix for all services.
#
wsdlUrlPrefix = protocol + "://" + host + ":" + port + "/" + contextUri + "/";
```

Initiate the service

```
FolderService fs = new FolderService();
fs.Url = wsdlUrlPrefix + "com.opsware.folder/FolderService";
```

Invoke service methods

```
FolderRef root = fs.getRoot();
FolderVO vo = fs.getFolderVO(root);
```

Handle interface return types

```
            # In the API, FolderVO.getMembers returns an array of
# FNodeReference interfaces, but Web Services does not support
# interfaces. In the C# stub, the return type of
# FolderVO.members is Object[]. If a returned Object type will
# be used as a parameter that must be a specific type, then you
# must cast it to that type. For example, the following code
# casts elements of the returned array to FolderRef as
# appropriate.
#
Object[] members = vo.members;
for(int i=0;i<members.Length;i++)
{
```

```
Console.WriteLine("Got object: " + members[i].GetType().FullName + " --> " +
((ObjRef)members[i]).name);
       if(members[i] is FolderRef) {
       Console.WriteLine("I am a FolderRef: " +
       ((FolderRef)members[i]).name);
       }
}
```

Update or create a VO

```
            # When updating a VO, the changed attributes must be set in
# dirtyAttributes. (The VO passed to a create method has
# the same requirement.)
#
# Note: If you update a VO that was returned from a service
# method invocation, such as getFolderVO, then you must
# set the logChange attribute of the VO to false:
# vo.logChange = false;
#
# The following code changes the name of a folder.
#
Console.WriteLine("Changing name from " + vo.name +
" to yo_csharp.");
vo.name = "yo_csharp";
vo.dirtyAttributes = new String[]{"name"};
# Manually set dirty fields being changed.
#
vo = fs.update(folder, vo, true, true);
Console.WriteLine("Folder name changed to: " + vo.name);
```

Handle exceptions

```
            # .NET converts Web Services faults into SoapExceptions
# without trying to deserialize them into application
# exceptions first. As a result, your code cannot catch
# application exceptions. As a workaround, the C# stubs
# provided by SA include SOAPExceptionParser,
# a class that enables you to get information from
# SOAPExceptions. The following code shows how to get the
# exception name and error message by calling the getDetail
# method of SOAPExceptionParser.
#
try{
// Try to get a non-existent folder here.
} catch(SoapException e){
       SoapExceptionDetail detail =
       SoapExceptionParser.getDetail(e);
       Console.WriteLine("SoapExceptionDetail.name: " +
       detail.exceptionName);
       Console.WriteLine("SoapExceptionDetail.msg: " +
       detail.message);
...
}
```

Password security with C#

The FolderService method reads the user and password pair from the file App.config. The following shows an example of this method.

```
            User user = new User();
user.username = "user";
user.password = "password";
FolderService fs = new FolderService();
fs.Url = wsdlUrlPrefix + "com.opsware.folder/FolderService";
fs.user = user;
```

If you do not want to store the password in clear text in the App.config file, you can use the SecureUser class to encrypt the password. The SecureUser class uses the C# SecureString in .NET 2.0. Passwords are stored encrypted in a SecureString. Furthermore, the getPassword() method is only visible internally. SecureUser is a static class, so you only need to set your user name and password once or each time you switch users.

Each service retrieves the user name and password from SecureUser first and then its user member variable and then App.config, for backward compatibility. SecureUser takes either a String or a SecureString for the password. In either case, clients are responsible to clean up the password variable passed to the SecureUser.setUser() method.

At some point the password will need to be converted to a regular C# string in memory, which will only get freed when the next garbage collection occurs. Using SecureUser will only ensure internal password storage is secure.

The following example shows how to set the user name and password securely.

```
            SecureString passwd = new SecureString();
passwd.AppendChar('p');
passwd.AppendChar('a');
passwd.AppendChar('s');
passwd.AppendChar('s');
passwd.AppendChar('w');
passwd.AppendChar('d');
SecureUser.setUser("username", passwd); // that's it, no need to set up user
```

```
for each service.
passwd.Dispose(); // resets passwd and frees up memory so no copy remains from
caller.
```

Pluggable checks

The SA Audit and Remediation feature enables you to define and monitor the compliance information for SA managed servers. Because compliance standards are continuously evolving, SA lets you create specialized custom checks and policies, and extend those provided with SA. A pluggable check is an audit rule, which belongs to one or more audit policies. You create a pluggable check in a command-line environment, upload the check, and then add it to an audit policy with the SA Client.

This section is intended for software developers who are familiar with XML and with the Audit and Remediation feature of SA.

Setup for pluggable checks

Before developing pluggable checks:

1. Install an SA core in a development environment. Do not use a production core.
2. On a server that has an installed Agent, install OCLI 1.0. For information on the OCLI 1.0, see Use.

Pluggable check tutorial

This tutorial shows how to create a pluggable check named HelloWorld Check. This simple check verifies that the `/var/tmp/helloworld` file exists on a Unix managed server. If the file does not exist, the remediation script of the pluggable check creates the file.

To develop the HelloWorld check:

1. Follow the instructions in Setup for Pluggable Checks. The server where you install OCLI 1.0 will be the development server for this tutorial.
2. The HelloWorld Check example code is included with the ZIP file that contains the API code examples.
3. Unzip the file you downloaded in the preceding step and verify that the `pluggable_checks/helloworld` directory contains the following files:
   - `config.xml`
   - `gethelloworld.py`
   - `sethelloworld.py`

   The HelloWorld check is made up of these three files. The `config.xml` file is a configuration file. The `gethelloworld.py` Python script performs the audit. The `sethelloworld.py` Python script performs the remediation. In the following steps, you package these files into a ZIP file and then import the ZIP file into SA.
4. On your development server, copy the unzipped `helloworld` files to a working directory, for example:

   ```
   cd /home/jdoe/dev
   mkdir helloworld
   cd helloworld
   cp unzip_dest/pluggable_checks/helloworld/* .
   ```
5. Obtain a Globally Unique ID (GUID). Each pluggable check requires a GUID. You can acquire a valid GUID by using one of the following techniques:
   - Log on to web sites such as the following:
     `http://kruithof.xs4all.nl/uuid/uuidgen`
   - Download the free Windows tool `guidgen` from:
     `http://www.microsoft.com/downloads/details.aspx?FamilyID=94551F58-484F-4A8C-`

```
BB39-ADB270833AFC&displaylang=en
```

If you programmatically create your GUIDs, then your code should conform to RFC4122 (`http://www.ietf.org/rfc/rfc4122.txt`).

6. With a text editor, insert the GUID in the `config.xml` file, for example:

   ```
   <checkGUID>6c7ed38c-d8d6-11db-8314-0800200c9a66</checkGUID>
   ```

   This is the only element in `config.xml` that you need to modify for this tutorial.

7. In the text editor, save `config.xml` with the change you made for the GUID.

   Keep the text editor open. Throughout this tutorial, you will examine various elements in `config.xml` to learn how they map to the Python scripts and the SA Client display fields of the HelloWorld Check.

8. In the `config.xml` file, note the following elements, which are related to the audit (get) and remediation (set) scripts of the HelloWorld Check:

   ```
   <!-- The name of the script that performs the check. -->
   <checkGetScriptName>gethelloworld.py</checkGetScriptName>

   <!-- The name of the script that remediates the audit. -->
   <checkSetScriptName>sethelloworld.py</checkSetScriptName>

   <!-- The exit code of the gethelloworld.py script will be checked.-->
   <checkReturnType>EXITCODE</checkReturnType>

   <!-- A string argument is passed to gethelloworld.py. -->
   <checkGetArgumentType>STRING</checkGetArgumentType>

   <!-- The default argument for gethelloworld.py is the name of the file the
   script is checking for. -->
   <checkGetArgumentDefaultValue>/var/tmp/helloworld
   </checkGetArgumentDefaultValue>

   <!-- If the helloworld file exists, the exit code of gethelloworld.py is 0.
   -->
   <checkSuccessExitCodeValue>0</checkSuccessExitCodeValue>
   <!-- If the helloworld file does not exist, the exit code of
   gethelloworld.py is 1. -->
   <checkSuccessExitCodeValue>1</checkSuccessExitCodeValue>
   ```

9. Examine the `gethelloworld.py` script, which performs the audit by checking for the existence of

the file /var/tmp/helloworld. You do not need to edit this script for this tutorial. Later in this tutorial (step 30, when you run the audit in the SA Client, this script executes on a managed server.

The /var/tmp/helloworld string is the default argument of the script, as indicated by the value of <checkGetArgumentDefaultValue> in config.xml. The script's exit code (result) corresponds to the values specified for <checkSuccessExitCodes>.

Here is the source code for the gethelloworld.py script:

```
import sys
import os
import string
if __name__ == "__main__":
       if len(sys.argv) != 2:
               sys.stderr.write("No argument found! Please enter a
                       file name!\n")
               sys.exit(220)
filename = sys.argv[1]
if os.path.isfile(filename) or os.path.isdir(filename):
       result = 0
else:
       result = 1
sys.stderr.write("Debugging: Found result %s\n"
% result)
sys.stdout.write("%s\n" % result)
sys.exit(result)
```

10. Next, examine the remediation script sethelloworld.py, which creates the /var/tmp/helloworld file. This script runs on a managed server if you decide to remediate the audit in step 35. Do not change the script for this tutorial.

    The source code for sethelloworld.py follows:

```
import sys
import os
import string
if __name__ == "__main__":
       if len(sys.argv) != 2:

                sys.stderr.write("No argument found!
                       Please enter a file name!\n")
               sys.exit(220)
filename = sys.argv[1]
if os.path.isfile(filename) or os.path.isdir(filename):
       # Do nothing because the file already exists.
```

```
        pass
else:
        try:
                fd = open(filename, "w")
                fd.write(" ")
                fd.close()
        except:
                sys.stderr.write("Could not open file %s for
                        writing!\n" % filename)
                sys.exit(220)
# Exit successfully with a 0 exit code.
sys.stderr.write("Successfully created file\n")
sys.exit(0)
```

11. Package the HelloWorld Check.

    To package the HelloWorld pluggable check, archive the contents of the working directory into a single ZIP file, for example:

    ```
    cd /home/jdoe/dev/helloworld
    zip ../helloworld.zip *
    ```

12. Verify that the ZIP file contains the two Python scripts and the `config.xml` file by entering the following `unzip` command:

    ```
    unzip -t ../helloworld.zip
    testing: config.xml OK
    testing: gethelloworld.py OK
    testing: sethelloworld.py OK
    No errors detected in compressed data of ../helloworld.zip.
    ```

13. Import the pluggable check into SA with the `oupload` command of OCLI 1.0:

    ```
    oupload -C"Customer Independent" \
    -t"Server Configuration Check" \
    --forceoverwrite --old -O"SunOS 5.8" ../helloworld.zip
    ```

    > **Note:** The platform option (`-O`) is `SunOS 5.8` for all Unix and Linux checks. For Windows checks, the platform option is `Windows 2003`.

    If `oupload` does not run successfully, make sure that you have installed the correct version of OCLI 1.0, set the PATH environment variable correctly, and included the `login` file in your environment. For details on these requirements, see the OCLI 1.0 in the Using

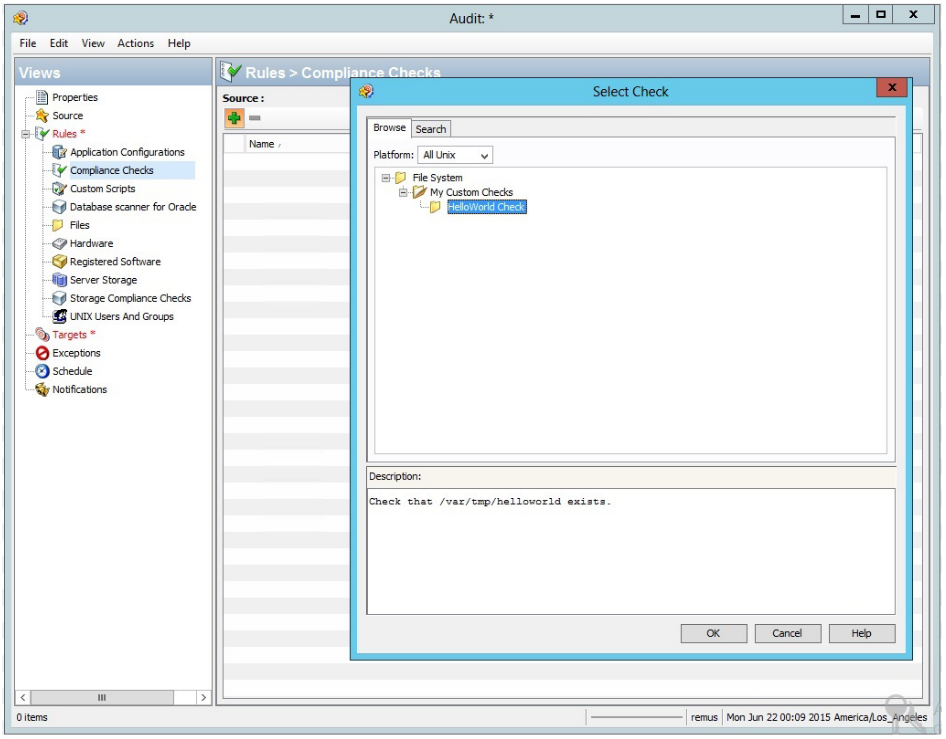
14. Open the SA Client.

In the next few steps, you create a new audit, adding to it the HelloWorld Check you imported with the `oupload` command.

15. From the **Tools** menu, select **Update Cache**.
16. From the Navigation pane, select **Library> By Type> Audits and Remediation > Audits> Unix**.
17. From the **Actions** menu, select **New**.
18. In the Audit Window, in the Name field of the Properties pane, enter HelloWorld Audit.
19. In the Views pane, In the Views pane, select **Rules** > **Compliance Checks**.
20. Click the **Add** button ✚, and then click **File System**.

    The Content pane should list the HelloWorld Check under Available for Audit, as shown in the following figure:

HelloWorld check in the rules for a file system



Platform: All Unix

Description:

Check that /var/tmp/helloworld exists.

21. In the `config.xml` file, note the following elements, which are related to the information displayed.

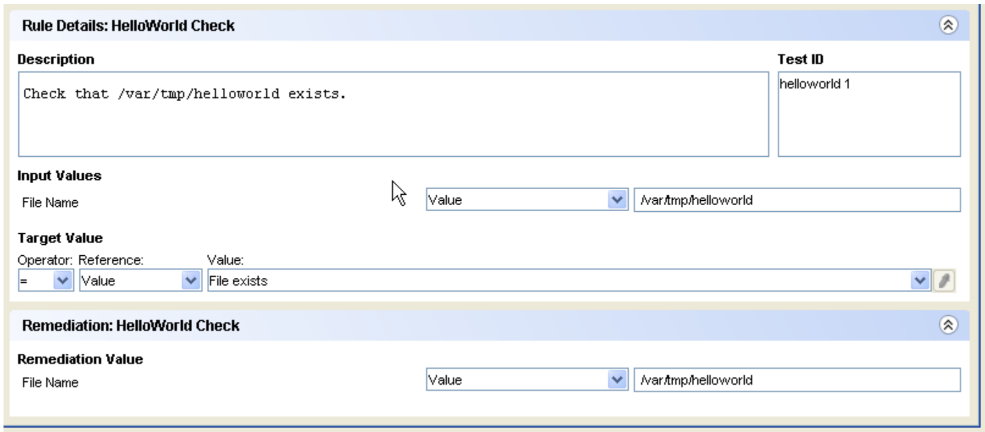
```
<!-- The check name is the rule name shown in the SA Client. -->
<checkName>HelloWorld Check</checkName>
```

```
<!-- The category corresponds to the rule hierarchy displayed by the SA
Client. -->
<checkCategory>File System|My Custom Checks</checkCategory>
```

In the Audit Window of the SA Client, under Available for Audit, select HelloWorld Check and click the plus sign.

The Content pane should list the details for HelloWorld Check, as shown in the following figure.

HelloWorld check rule details



Rule Details: HelloWorld Check

Description: Check that /var/tmp/helloworld exists.

Test ID: helloworld 1

Input Values

File Name: Value /var/tmp/helloworld

Target Value

Operator: = Reference: Value Value: File exists

Remediation: HelloWorld Check

Remediation Value

File Name: Value /var/tmp/helloworld

22. In the `config.xml` file, examine the following elements, which are related to the information displayed under the "HelloWorld check rule details" above figure:

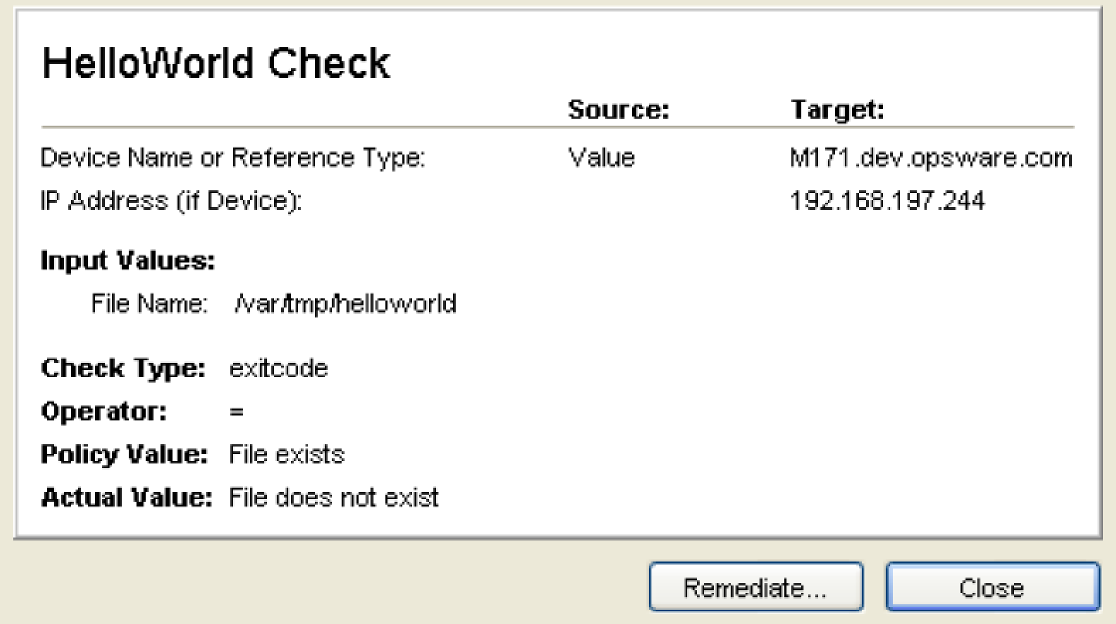
```
<!-- The following value appears under Description in the Rule Details of
the SA Client. -->
<checkDefaultDescription>
Check that /var/tmp/helloworld exists.
</checkDefaultDescription>

<!-- The following element corresponds to the Test ID in the SA Client. -->
<checkTestID>helloworld 1</checkTestID>

<!-- This label is under Input Values in the SA Client. -->
<checkGetArgumentDefaultLabel>File Name
</checkGetArgumentDefaultLabel>

<!-- The default argument to the gethelloworld.py script also appears
under Input Values in the SA Client. -->
<checkGetArgumentDefaultValue>/var/tmp/helloworld
</checkGetArgumentDefaultValue>
```

23. In the Views pane of the SA Client, select Targets.
24. In the following steps you add a target server to HelloWorld Audit. In later steps, the `gethelloworld.py` and `sethelloworld.py` scripts will run on the target server.
25. In the Contents pane, click **Add**.
26. In the Select Server window, drill down to a server and click **OK**.
27. In the Audit window, select **File> Save**.

    At this point, the HelloWorld Audit contains the HelloWorld Check (rule) and is associated with a target server.
28. In the Audit window, from the **Actions** menu, select **Run Audit**.
29. Step through the windows of the Run Audit task.
30. In the Run Audit window, click **Start Job**.
    This action launches the job that runs the `gethelloworld.py` script on the target server.
31. After the job has completed, click **View Results**.
32. In the Views pane of the Audit Result window, select Policy Rules (1).
33. In the Content pane of the Audit Result window, open HelloWorld Check.

    The Difference Details window should appear, as shown in the following figure.

    HelloWorld check difference details



HelloWorld Check

	Source:	Target:
Device Name or Reference Type:	Value	M171.dev.opsware.com
IP Address (if Device):		192.168.197.244

Input Values:

File Name: /var/tmp/helloworld

Check Type: exitcode

Operator: =

Policy Value: File exists

Actual Value: File does not exist

34. In the `config.xml` file, note the following elements, which are related to the information displayed in the "HelloWorld check difference details" on the previous page figure:

    ```
    <!-- The following value appears as the Policy Value in the Difference
    Details window. -->
    <checkSuccessExitCodeDefaultDisplayName>
    File exists</checkSuccessExitCodeDefaultDisplayName>


    <!-- The next value appears as the Actual Value in the same window. -->
    <checkSuccessExitCodeDefaultDisplayName>
    File does not exist</checkSuccessExitCodeDefaultDisplayName>
    ```

35. If you want to create `/var/tmp/helloworld` on the target server, on the Differences Window, click **Remediate**.

    This action runs the `sethelloworld.py` script. For more information, see the Server Automation Administration Guide on the HPE SSO portal.

Audit and remediation

Sarbanes-Oxley (SoX), Information Technology Infrastructure Library (ITIL), and ISO20000 make it urgent to keep server configurations in compliance. The SA Audit and Remediation feature offers you a well-organized set of policies to help you address compliance issues. A graphical interface makes it easy for you to select and run audits against specified servers, and see how well they comply with professional standards.

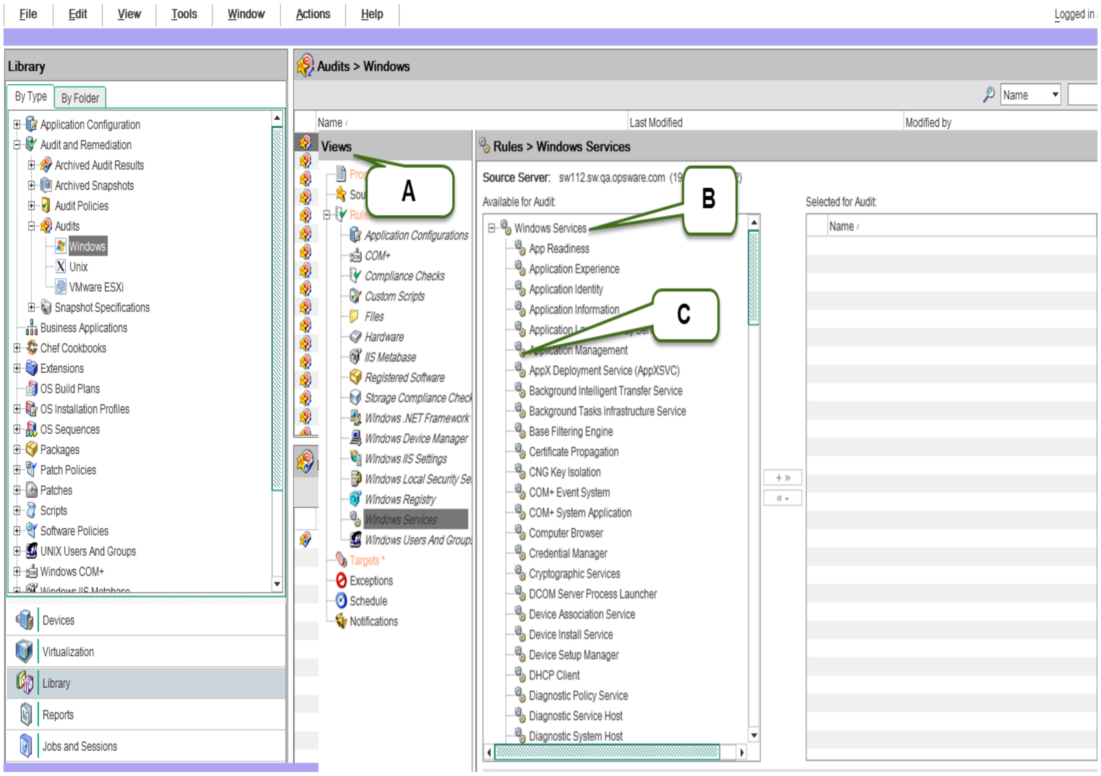
Audit and Remediation also simplifies system administration. For example, you might monitor a class of servers that run a home grown application built by your team, such as a database server or middleware application. As you configure and monitor the servers that run the application, you keep a list that tracks the ideal state of the configuration. Such a list might include file, directory, and network share permissions.

You can create an audit that defines these configurations, then audit the servers after installing the application. The audit results will confirm whether or not the application is installed and has been configured successfully according to your criteria. If the configuration is non-compliant, you can create an ad-hoc audit to troubleshoot the problem. When the audit results indicate an error, you can remediate the server to match your ideal configuration. To ensure that the configuration change works in production, you can set the audit to run on a configurable schedule and have a notification sent upon completion.

Showing a window for selecting an audit, the following figure includes the following callouts:

- **Callout A**: Any category listed in the Views panel may have SA non-modifiable capabilities, or modifiable pluggable checks.
- **Callout B**: This points to the SA capabilities for dealing with Windows services.
- **Callout C**: This lists pluggable checks for working with Windows Services.

Windows Services audit rule



Each check evaluates one rule. Several checks can be bundled together into a policy.

The SA Audit and Remediation feature comes with many out-of-the-box checks. You can run most audits by selecting the desired check. The choice of audits grows continuously as developers design, code, test, and add more checks to the system through the HPE Live Network. These checks are imported as complete policies.

However, since every business has unique challenges and unique resources, you may need to determine compliance against a set of criteria not available for auditing within the SA Audit and Remediation framework. For this reason, the system provides a way to create your own custom pluggable checks.

The Audit and Remediation feature evaluates, by specific rules, the compliance state of servers under SA management. This feature can also remediate the servers that do not match the desired configuration state as defined in the rules. These rules include various server parameters, registry values, file permissions, application configurations, file existence, COM+ objects, and more.

In the Windows environment, web server rules can also be specified with application configuration, which is based upon the Microsoft Internet Information Services (IIS) Web server configuration file, UrlScan.ini. SA can compare partial or full values from specific configuration files, select the desired elements from the file, and make sure that these values or configuration file entries exist. For more information, see the Application Configuration.

SA includes many predesigned audit rules. Each defines a desired state of configuration for a server or server groups. Some rules are value-based, providing a comparator (<, >, ==, !=, contains, etc.), a value or set of values, and one or more checks, which spell out the underlying code used to evaluate the state of the audited item or items. The comparison data determines compliance or non-compliance. A rule may also contain remediation values if the check supports remediation.

A rule consists of a single check. You can create new functionality by using custom content objects in the form of pluggable checks. You can also bundle related pluggable checks into audit policies for convenience.

Creating a pluggable check

A pluggable check is code that is downloaded to the managed server or servers and is executed by the Audit and Remediation framework. You can use checks to extend the native Audit and Remediation properties and to provide additional specialized functionality. Each pluggable check includes a customized config.xml file and at least one script that compares the audited feature against values specified in the config.xml file. A pluggable check may also include a script that sets specified variables in the audited server to the value specified in the config.xml file. You can write pluggable check scripts in Python, Visual Basic Scripting (VBS), BAT, or shell script. A pluggable check is packaged as a zip archive.

Most of the CIS checks are direct translations of the CIS benchmarks. More information can be found at http://www.cisecurity.org.

Most types of checks fall into one of the following categories:

- Windows Registry checks
- Unix Services checks
- User checks, which may use password or shadow file information

Guidelines for pluggable checks

To simplify server maintenance, adhere to the following guidelines:

- When creating a new pluggable check, pay special attention to the names. Describe the purpose of the check, and replace spaces with an underscore. For example, `Users_Without_Password_Expiration` is self-explanatory. This will help you to find a check quickly when a server acquires several hundred or more checks.
- Write a generic check. This enables you to easily create additional checks of the same execution type with only a few lines of code change. For example, for most CIS2k3 Windows Service Checks, you can change a single line of code to create a new check for a new service.
- When naming the audit (get) and remediation (set) scripts, remove the spaces or underscores from the directory name, and prefix with get or set, as appropriate. For example, `getUsersWithoutPasswordExpiration.sh` is a good name for an audit file. Be consistent on this, even if you think your custom check will not be used by anyone else.

- Pay attention to error checking. Remember that unexpected return values might report an audit as non-compliant when a script failure occurs. Trap the unexpected error or exception, and write out information about it to stdout or stderr to simplify troubleshooting.
- Convert most checks to a simple binary case of True or False when possible.
- Always try to handle not only the specific benchmark case, but also its counterpart. For example, you can easily create a “Disable Service X,” pluggable check at the same time that you create an “Enable Service X” and reuse most of the code. This can be useful if you decide later to test for the opposite condition.
- Use the standard exit codes defined by the framework whenever possible. These are:

  ```
  EXIT_FAILURE=220
  EXIT_ERR_USAGE=221
  EXIT_ERR_INVALID_OS=222
  ```

- When returning disabled or enabled in a Boolean type check, return 0 for disabled, 1 for enabled.
- Package each pluggable check as a ZIP archive. A single file system directory contains the files listed in the following table.

Pluggable check contents

File Name	Description
`config.xml`	(Required*)* The XML configuration file defining how this pluggable check executes, returns, and ultimately reports compliance or non-compliance*.*
`get`*`Name`*`. {py | sh | BAT | vbs}`	(Required*)* The audit script, written in Python, VBS, BAT, or shell, that evaluates the audited object, and returns text and exit codes according to the config.xml definitions.
`set`*`Name`*`. {py | sh | BAT | vbs}`	(Optional) The remediation script, written in Python, VBS, BAT, or shell, that remediates the condition checked by the audit script.
Additional Code, Scripts, or Libraries	(Optional) Helper and supplementary scripts used by either the audit or remediation scripts*.*

The file names for the audit and remediation scripts do not need to begin with get and set, but this convention simplifies file maintenance.

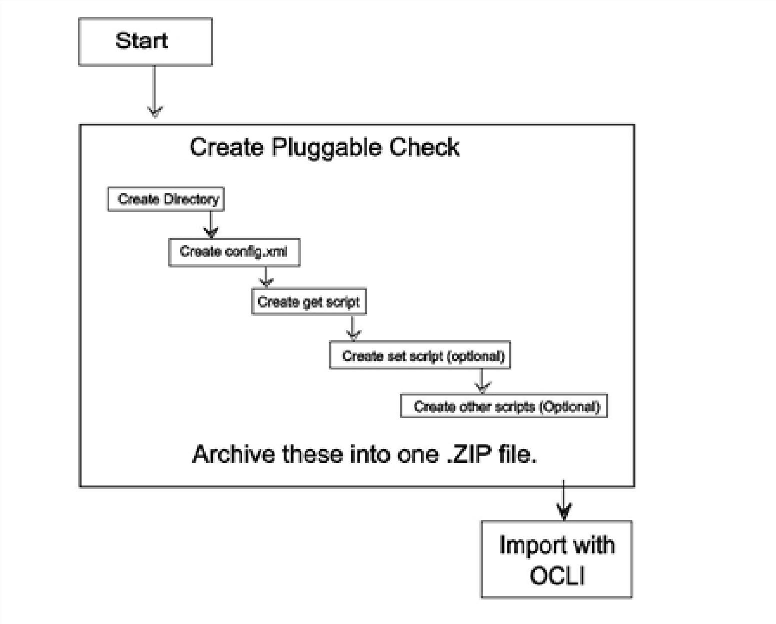
The following example shows a directory structure for a pluggable check:

```
./check_name/
./check_name/config.xml
./check_name/getcheckname.py
./check_name/setcheckname.py
```

Development process for pluggable checks

The following figure shows an overview for the development process, which takes place in a command-line environment.

Development process



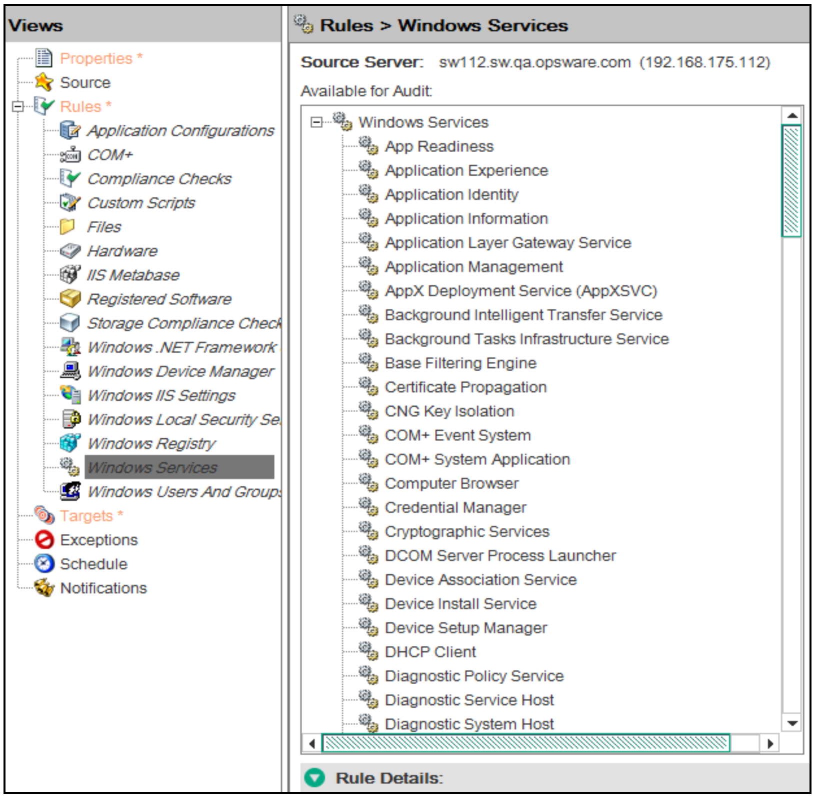
Pluggable check configuration (config.xml)

The config.xml file is a specification file for the pluggable check that contains elements to control how this check appears in the SA Client, default values, value types for comparison, and the category of the check. For example, the following element in the `config.xml` file determines the pluggable check's rule category in the SA Client:

```
<checkCategory>Windows Services</checkCategory>
```

Standard categories, each indicated with its own icon, include hardware, software, operating systems, users and groups, file systems, and more, as shown in the following figure.

Pluggable check categories in the rule hierarchy



Source Server: sw112.sw.qa.opsware.com (192.168.175.112)

The following listing shows the template for the `config.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE checkConfiguration SYSTEM "check.dtd">
<checkConfiguration version="1.0">
<checkName>$CHECKNAME</checkName>
<checkGUID>$CHECKGUID</checkGUID>
<checkDefaultDescription>$CHECKDESCRIPTION</checkDefaultDescription>
<checkRemediationDefaultDescription> $CHECKREMEDIATIONDESCRIPTION </
checkRemediationDefaultDescription>
<checkGetScriptName>$GETSCRIPTNAME</checkGetScriptName>
<checkGetScriptType>PY</checkGetScriptType><!-- Or SH for shell, BAT for Bat,
VBS for Visual Basic -->
<checkSetScriptName>$SETSCRIPTNAME</checkSetScriptName><!-- Optional -->
<checkSetScriptType>PY</checkSetScriptType><!-- Optional -->
<checkVersion>32b.0-1.0</checkVersion>
<checkReturnType>$RETURNTYPE</checkReturnType> <!-- EXITCODE, STRING, or
NUMBER -->
<checkTestIDs>
<checkTestID>$CHECKTESTID</checkTestID> <!-- Optional -->
</checkTestIDs>
<checkPlatformTypes>
<checkPlatform>$PLATFORMTYPE</checkPlatform> <!-- Currently Unix or Windows --
>
</checkPlatformTypes>
```

```
<checkCategories>
<checkCategory>$CATEGORY</checkCategory> <!-- Top-level GUI category -->
</checkCategories>
<checkGetArguments> <!-- All arguments are optional -->
<checkGetArgument>
<checkGetArgumentType>$GETARGTYPE</checkGetArgumentType> <!-- STRING or NUMBER
-->
       <checkGetArgumentDefaultLabel>$GETDEFAULTLABEL</
checkGetArgumentDefaultLabel>
               <checkGetArgumentDefaultDescription>$GETDEFAULTDESCRIPTION</
checkGetArgumentDefaultDescription>
               <checkGetArgumentDefaultValue>$GETDEFAULTVALUE</
checkGetArgumentDefaultValue>
       </checkGetArgument>
</checkGetArguments>
<checkSetArguments> <!-- Also optional -->
<checkSetArgument>
<checkSetArgumentType>$SETARGTYPE</checkSetArgumentType>
       <checkSetArgumentDefaultLabel>$SETDEFAULTLABEL</
checkSetArgumentDefaultLabel>
               <checkSetArgumentDefaultDescription>$SETDEFAULTDESCRIPTION</
checkSetArgumentDefaultDescription>
       <checkSetArgumentDefaultValue>$SETDEFAULTVALUE</
checkSetArgumentDefaultValue>
</checkSetArgument>
</checkSetArguments>
<checkSuccessExitCodes> <!-- Only for EXITCODE type checks, generally at least
two entries -->
       <checkSuccessExitCode>
<checkSuccessExitCodeValue>$EXITCODEVALUE</checkSuccessExitCodeValue>
               <checkSuccessExitCodeDefaultDescription>$EXITCODEDESCRIPTION
       </checkSuccessExitCodeDefaultDescription>
               <checkSuccessExitCodeDefaultDisplayName>$EXITCODEDISPLAYNAME
       </checkSuccessExitCodeDefaultDisplayName>
</checkSuccessExitCode>
</checkSuccessExitCodes>
</checkConfiguration>
```

For more details, see Document Type Definition (DTD) for config.xml File.

Audit (get) scripts

You can design the audit script, also known as the get script, to obtain a value from a managed server. The script is executed with optional parameters, as specified in the `config.xml` file. If the script is running an EXITCODE check, the result of the script is compared to the exit codes specified in the config.xml file. For STRING and NUMBER return type checks, the result is compared to what is written to STDOUT.

An audit script has a set of pre-defined return codes. You can define additional return codes in the check config.xml file.

The audit script may display informational messages. These messages are useful when troubleshooting an audit script failure. Review the following sample Python audit script:

Remediation (set) scripts

You can design the remediation script, also known as the set script, to enact a change on the managed server that would cause the audit script to return success when completed. The script is executed with optional parameters, as specified in the check config.xml file.

These set scripts are optional, and can vary in character from being very similar to their counterpart get scripts to entirely different (and longer).

From a shell standpoint, there is nothing special in the script itself, other than the return codes being used. Most checks display some debug output or information messages. This is not normally seen by users, except in the event of a script failure, where the messages are useful for troubleshooting purposes.

As a standard practice, always include at least one parameter to the set script. Also, remember to modify the config.xml file so that it displays nicely in the SA Client when adding a set script to an already existing check.

Make sure your remediation scripts exit with exit code 0 to indicate success. All other exit codes will indicate failure of the remediation operation.

Review the following sample Python set script.

```
import sys
import os
import string
if __name__ == "__main__":
# If there are set arguments they will be loaded into
# sys.argv
# Enter the desired set code here. Stdout may be used for
# debugging.
# Uses exitcode 0 for success, and all other values for
# failure.
# enter condition where set script if successful. for this
# example, use ‘if 1’
if 1:
	sys.exit(0)
else:
	sys.exit(-1)
```

Other code for pluggable checks

Pluggable checks may also contain code other than the get or set scripts. Libraries, executables or additional scripts can be added to the check, so their set or get scripts can utilize these upon execution.

You can also include additional code in the ZIP file.

Zipping up pluggable checks

After you have created the `config.xml` file, the audit (get) script, and the optional remediation (set) script, create a ZIP archive containing these files. The following shell history shows the creation process in a UNIX environment.

```
# ls
check_name
# cd check_name
# zip ../checkname.zip *
adding: config.xml
adding: getcheckname.py
adding: setcheckname.py
# unzip -t ../checkname.zip
testing: config.xml OK
testing: getcheckname.py OK
testing: setcheckname.py OK
No errors detected in compressed data of ../checkname.zip.
```

Importing pluggable checks

Import a pluggable check into an SAcore or mesh using the OCLI 1.0 utility, which is documented in the SA Content Utilities. The following shell history provides an example of the import process for Linux:

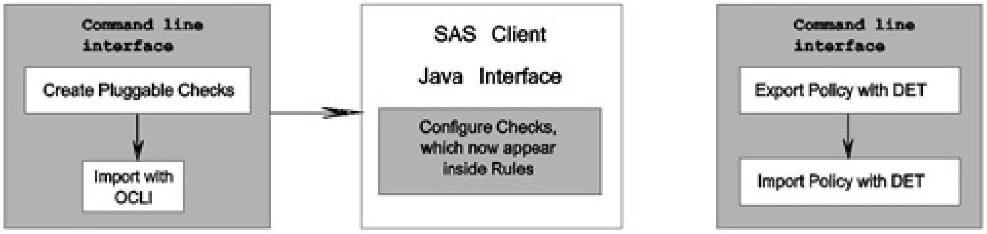
```
# cp checkname.zip /var/tmp/checks
# cd /var/tmp/checks
# cp opsware_32.a.692.0-upload/disk001/packages/Linux/3AS/ocli-32a.2.0.5-
linux-3AS .
# chmod 755 ocli-32a.2.0.5-linux-3AS
# ./ocli-32a.2.0.5-linux-3AS
# . ./ocli/login.sh
# export PATH=/opt/opsware/bin:$PATH
# oupload -C"Customer Independent" -t"Server Configuration Check" --
forceoverwrite --old -O"SunOS 5.8" your_Pluggable_check.zip
```

The `oupload` command uses "SunOS 5.8" to specify that the check falls into the generic Unix category in the SA Client. To specify a check for the Windows category, use "Windows 2003."

Creating the audit policy

The audit policy creation procedure is illustrated in figure below:

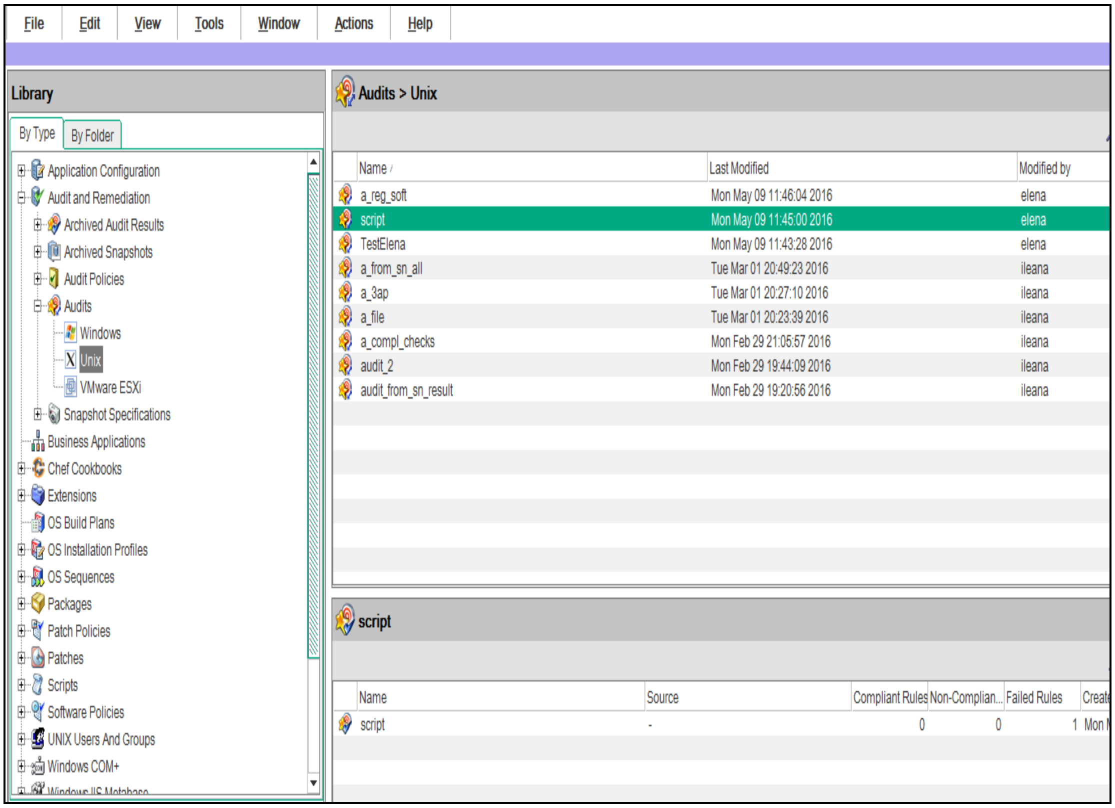
Procedure for creating an audit policy



Creating an audit policy

Audit policies consist of rules. Each rule consists of one or more checks, which can include the user-created pluggable check. Audit policies and rules are displayed, created and edited in the SA Client. The following figure shows a list of the audit rules available on a model system.

List of audit rules



Name	Last Modified	Modified by
a_reg_soft	Mon May 09 11:46:04 2016	elena
script	Mon May 09 11:45:00 2016	elena
TestElena	Mon May 09 11:43:28 2016	elena
a_from_sn_all	Tue Mar 01 20:49:23 2016	ileana
a_3ap	Tue Mar 01 20:27:10 2016	ileana
a_file	Tue Mar 01 20:23:39 2016	ileana
a_compl_checks	Mon Feb 29 21:05:57 2016	ileana
audit_2	Mon Feb 29 19:44:09 2016	ileana
audit_from_sn_result	Mon Feb 29 19:20:56 2016	ileana

Name	Source	Compliant Rules	Non-Complian...	Failed Rules	Create
script	-	0	0	1	Mon N

For detailed information on creating an audit policy, see the Server Automation Administration Guide on the HPE SSO portal.

Exporting the audit policy

To move a new audit policy to other SA cores, export it from one and import it to another using the DCML Exchange Tool (DET) command-line utility. Use this tool to populate a newly-installed SA core with content, such as policies, from an existing core. For detailed instructions on this procedure, see the Server Automation Administration Guide on the HPE SSO portal.

Document Type Definition (DTD) for config.xml file

This file governs SA Client display names and descriptions, default values, comparisons to be performed upon values returned by the check code, the category of the SA Client displaying these values, and more.

Two elements in the default `config.xml` file, `checkGetArguments` and `checkSetArguments`, are used to pass data values to the scripts at execution time. If your programmable check does not require any arguments, delete these elements from your `config.xml` file.

The following DTD for `config.xml` is dynamically generated by SA:

```
<!ELEMENT checkConfiguration (checkName, checkGUID, checkDefaultDescription,
checkRemediationDefaultDescription?, checkGetScriptName?,
checkGetScriptType?, checkSetScriptName?, checkSetScriptType?, checkVersion,
checkAllowRemediationOnFailure?, checkReturnType, checkTestIDs?,
checkPlatformTypes, checkExclusivePlatforms?, checkExcludePlatforms?,
checkCategories, checkGetArguments?, checkSetArguments?,
checkComparisonDefaults?, checkCompareValidValues?, checkSuccessExitCodes?)>
<!ATTLIST checkConfiguration version CDATA #REQUIRED>
<!ELEMENT checkName (#PCDATA)>
<!ELEMENT checkGUID (#PCDATA)>
<!ELEMENT checkDefaultDescription (#PCDATA)>
<!ELEMENT checkRemediationDefaultDescription (#PCDATA)>
<!ELEMENT checkGetScriptName (#PCDATA)>
<!ELEMENT checkGetScriptType (#PCDATA)>
<!ELEMENT checkSetScriptName (#PCDATA)>
<!ELEMENT checkSetScriptType (#PCDATA)>
<!ELEMENT checkVersion (#PCDATA)>
<!ELEMENT checkAllowRemediationOnFailure (#PCDATA)>
<!ELEMENT checkReturnType (#PCDATA)>
<!ELEMENT checkTestIDs (checkTestID+)>
<!ELEMENT checkTestID (#PCDATA)>
<!ELEMENT checkPlatformTypes (checkPlatform+)>
<!ELEMENT checkPlatform (#PCDATA)>
<!ELEMENT checkExclusivePlatforms (checkExclusivePlatform+)>
<!ELEMENT checkExclusivePlatform (#PCDATA)>
<!ELEMENT checkExcludePlatforms (checkExcludePlatform+)>
<!ELEMENT checkExcludePlatform (#PCDATA)>
<!ELEMENT checkCategories (checkCategory+)>
<!ELEMENT checkCategory (#PCDATA)>
<!ELEMENT checkGetArguments (checkGetArgument+)>
<!ELEMENT checkGetArgument (checkGetArgumentType,
checkGetArgumentDefaultLabel, checkGetArgumentDefaultDescription,
checkGetArgumentDefaultValue?, checkGetArgumentValidValues?)>
<!ELEMENT checkGetArgumentType (#PCDATA)>
<!ELEMENT checkGetArgumentDefaultLabel (#PCDATA)>
<!ELEMENT checkGetArgumentDefaultDescription (#PCDATA)>
<!ELEMENT checkGetArgumentDefaultValue (#PCDATA)>
<!ELEMENT checkGetArgumentValidValues (checkGetArgumentValidValue+)>
<!ELEMENT checkGetArgumentValidValue (checkGetArgumentValidValueItem,
checkGetArgumentValidValueDisplayName)>
<!ELEMENT checkGetArgumentValidValueItem (#PCDATA)>
<!ELEMENT checkGetArgumentValidValueDisplayName (#PCDATA)>
<!ELEMENT checkSetArguments (checkSetArgument+)>
<!ELEMENT checkSetArgument (checkSetArgumentType,
checkSetArgumentDefaultLabel, checkSetArgumentDefaultDescription,
checkSetArgumentDefaultValue?, checkSetArgumentValidValues?)>
<!ATTLIST checkSetArgument populateFromRule CDATA #IMPLIED>
<!ELEMENT checkSetArgumentType (#PCDATA)>
<!ELEMENT checkSetArgumentDefaultLabel (#PCDATA)>
<!ELEMENT checkSetArgumentDefaultDescription (#PCDATA)>
<!ELEMENT checkSetArgumentDefaultValue (#PCDATA)>
<!ELEMENT checkSetArgumentValidValues (checkSetArgumentValidValue+)>
<!ELEMENT checkSetArgumentValidValue (checkSetArgumentValidValueItem,
checkSetArgumentValidValueDisplayName)>
<!ELEMENT checkSetArgumentValidValueItem (#PCDATA)>
<!ELEMENT checkSetArgumentValidValueDisplayName (#PCDATA)>
<!ELEMENT checkComparisonDefaults (checkComparisonDefaultOperator?,
checkComparisonDefaultValues)>
<!ELEMENT checkComparisonDefaultOperator (#PCDATA)>
<!ATTLIST checkComparisonDefaultOperator not CDATA #IMPLIED>
<!ATTLIST checkComparisonDefaultOperator caseInsensitive CDATA #IMPLIED>
<!ELEMENT checkComparisonDefaultValues (checkComparisonDefaultValue+)>
<!ELEMENT checkComparisonDefaultValue (checkComparisonDefaultValueItem,
checkComparisonDefaultValueDisplayName)>
<!ELEMENT checkComparisonDefaultValueItem (#PCDATA)>
<!ELEMENT checkComparisonDefaultValueDisplayName (#PCDATA)>
<!ELEMENT checkCompareValidValues (checkCompareValidValue+)>
<!ELEMENT checkCompareValidValue (checkCompareValidValueItem,
checkCompareValidValueDisplayName)>
<!ELEMENT checkCompareValidValueItem (#PCDATA)>
<!ELEMENT checkCompareValidValueDisplayName (#PCDATA)>
<!ELEMENT checkSuccessExitCodes (checkSuccessExitCode+)>
<!ELEMENT checkSuccessExitCode (checkSuccessExitCodeValue,
checkSuccessExitCodeDefaultDescription,
checkSuccessExitCodeDefaultDisplayName)>
<!ELEMENT checkSuccessExitCodeValue (#PCDATA)>
<!ELEMENT checkSuccessExitCodeDefaultDescription (#PCDATA)>
<!ELEMENT checkSuccessExitCodeDefaultDisplayName (#PCDATA)>
```

The following table describes the elements of the `config.xml` DTD.

DTD Elements and Attributes

Elements	Attributes
`checkConfiguration version`	Set to 1.0, only change if the Audit and Remediation framework requires it.
`checkName`	The English name that displays in the SA Client for the check/rule.
`checkGUID`	A standard GUID, for example, 9500A4AE-EE9E-4383-87F2-BAD7DDC26C59 can be generated using the “guidgen” Windows utility, downloaded from a web site, or by other means. The GUID MUST be unique or the pluggable check will fail on upload to core. Once a check is uploaded with its unique GUID, you MUST NOT change the GUID or it will fail on re-upload with a "Database Unique Constraint Error" until you delete the original. Checks are uniquely identified by GUID, but for upload are solely identified by their name (of the zip file).
`checkDefaultDescription`	Displays in the SA Client description box. Honors hard carriage returns and HTML. With HTML, the HTML tags need to be converted with &lt; and &gt;.
`checkRemediationDefaultDescription`	Displays in the SA Client under the Remediation section of the check/rule.
`checkGetScriptName`	The file name for the get script, for example, getUsersWithoutPasswordExpiration.sh.
`checkGetScriptType`	The type of code determines the interpreter to be run. Get and set scripts may be types: SH, VBS, PY, BAT.
`checkSetScriptName`	The file name for the remediation script.
`checkSetScriptType`	The type of code determines interpreter to be run. Set (remediation) scripts may be of types SH, VBS, PY, BA.
`checkVersion`	This is based on SA and framework build number, such as 32b.0-1.0.

DTD Elements and Attributes, continued

Elements	Attributes
`checkAllowRemediationOnFailure`	Some scripts may fail during the get phase, but you may be able to correct this condition via the remediation script. This allows remediation to be performed even in the event of a script failure. For example, if the non-existence of a registry key is undefined, you can create and set it in your set code.
`checkReturnType`	Permissible values are EXITCODE, STRING, or NUMBER: EXITCODE — Standard script return via Wscript.Quit(), exit, return, etc. NUMBER — Audit and Remediation framework will grab from stdout and interpret it as numeric type. STRING — Audit and Remediation framework will grab from stdout and interpret as a string type.
`checkTestIDs`	List of test IDs.
`checkTestID`	Used to display the CIS, MSFT, NSA or other Policy standard nomenclature, for example, CIS-RHEL 8.4. This is a free form field, and displays in the SA Client, so be consistent in naming it to correspond with the TON Content.
`checkPlatformTypes`	List of valid platform types for a check.
`checkPlatform`	WINDOWS | UNIX (or both as individual elements)
`checkExclusivePlatforms`	List of exclusive platforms. Audit and Remediation currently separates things by Windows or Unix by default, but real world standards as well as limitations and/or differences across operating systems do not make this always desirable. You can limit Audit and Remediation to any platform specified by a platform ID retrieved from the spin. This parameter may refer to one of the supported operating systems listed in the SA Supported Platforms documentation.

DTD Elements and Attributes, continued

Elements	Attributes
`checkExclusivePlatform`	Individual platform ID.
`checkExcludePlatforms`	List of excluded platforms. If the PlatformType claims UNIX, you can supply platform IDs to exclude from the UNIX set (all Linux + all Unixes).
`checkExcludePlatform`	Individual platform ID
`checkCategory`	This is the SA Client Category that a check displays in. Currently, a check can only display in a single category. If a category does not exist, it will be created upon upload. The following standard categories for existing checks should be used where possible: Event Logging File System Operating System Operating System|Domain Controller (sub-category) Operating System|Network (sub-category) Registry Services Users and Groups
`checkGetArgument (checkGetArgumentType, checkGetArgumentDefaultLabel, checkGetArgumentDefaultDescription, checkGetArgumentDefaultValue?, checkGetArgumentValidValues?)>`	Specifies parameters to the get script.
`checkGetArgumentType`	NUMBER | STRING
`checkGetArgumentDefaultLabel`	SA Clienttag next to the input box or drop-down.
`checkGetArgumentDefaultDescription`	Hover text with further explanation.
`checkGetArgumentDefaultValue`	Default value for this get parameters.
`checkGetArgumentValidValue (checkGetArgumentValidValueItem, checkGetArgumentValidValueDisplayName`	checkGetArgumentValidValueItem (#PCDATA)> checkGetArgumentValidValueDisplayName (#PCDATA)>
`checkGetArgumentValidValues`	(Optional) Useful for limiting the parameters

DTD Elements and Attributes, continued

Elements	Attributes
`(checkGetArgumentValidValue+)`	for example to 0/disable and 1/enable.
`checkSetArguments (checkSetArgument+)`	checkSetArgument (checkSetArgumentType, checkSetArgumentDefaultLabel, checkSetArgumentDefaultDescription, checkSetArgumentDefaultValue?, checkSetArgumentValidValues?) setArgument elements are identical to the GetArguments, but for the remediation/set script if it exists. The exception is: checkSetArgument populateFromRule — the set parameter default should or should not populate itself from the rule data, versus if any default values were supplied in config.xml. Generally, this is always set to true.
`checkSetArgumentType`	NUMBER | STRING
`checkSetArgumentDefaultLabel`	SA Clienttag next to the input box or drop-down.
`checkSetArgumentDefaultDescription`	Hover text with further explanation.
`checkSetArgumentDefaultValue`	Default value for this set parameter.
`checkSetArgumentValidValues (checkSetArgumentValidValue+)`	
`checkSetArgumentValidValue (checkSetArgumentValidValue Item, checkSetArgumentValidValue DisplayName)>`	checkSetArgumentValidValueItem (#PCDATA)> checkSetArgumentValidValueDisplayName (#PCDATA)> checkSetArgumentValidValueItem (#PCDATA)> checkSetArgumentValidValueDisplayName (#PCDATA)>
`checkSetArgumentValidValue Item`	(Optional) Useful for limiting the parameters for example to 0/disable and 1/enable.
`checkSetArgumentValidValueDisplayName`	
`<!ELEMENT checkComparisonDefaults`	checkComparisonDefaultOperator not —

DTD Elements and Attributes, continued

Elements	Attributes
`(checkComparisonDefaultOperator?, checkComparisonDefaultValues)>`	negation of operator specified, TRUE | FALSE checkComparisonDefaultOperator caseInsensitive — only valid for STRING types.
`<!ELEMENT checkComparisonDefaultOperator (#PCDATA)>`	List of default values for comparator. Useful for field or development outside the TON build framework.
`checkComparisonDefaultValues (checkComparisonDefaultValue+)`	checkComparisonDefaultValue (checkComparisonDefaultValueItem, checkComparisonDefaultValueDisplayName).
`checkComparisonDefaultValueIte`	Value for default, passed to code.
`checkComparisonDefaultValueDisplayName`	Display name for the value, seen in the SA Client.

`checkCompareValidValues (checkCompareValidValue+)>`
`checkCompareValidValue (checkCompareValidValueItem, checkCompareValidValueDisplayName)>`
`checkCompareValidValueItem (#PCDATA)>`
`checkCompareValidValueDisplayName (#PCDATA)>`	

`checkSuccessExitCodes (checkSuccessExitCode+) checkSuccessExitCode (checkSuccessExitCodeValue, checkSuccessExitCodeDefaultDescription, checkSuccessExitCodeDefaultDisplayName)>`	For a checkReturnType of EXITCODE, you must define the valid values for proper script operation, which generally include both the compliant and non-compliant expected values. Anything returned other than a value specified here will be seen as a script failure, which is shown differently in the SA Client, as well as in reporting.
`checkSuccessExitCodeValue`	Value for script completion, for example, 0 (for *disabled* typically).
`checkSuccessExitCodeDefaultDescription`	Hover text for the DisplayName/Value.
`checkSuccessExitCodeDefaultDisplayName`	Value or text shown to user for this value, for example, Disabled.

Search filter syntax

Filter grammar

A search filter is a parameter for methods such as `findServerRefs`. The expression in a search filter enables you to get references to SA objects (such as servers and folders) according to the values of the object attributes. The formal syntax for a search filter follows:

```
<filter>                ::=     (<expression-junction>)+
<expression-junction>   ::=     <expression-list-open> <junction>
(<expression>)+ <expression-list-close>
<expression>            ::=     <expression-open> <attribute> <general-delimiter> <operator>
<general-delimiter> <value-list> <expression-close>

<attribute>             ::=     <resource_field>
<vo_member>             ::=     <text>
<resource_field>        ::=     <text>
<value-list>            ::=     (<double-quote> <text> <double-quote>)* |
(<number>)*
<text>                  ::=     [a-z] [A-Z] [0-9]
<number>                ::=     [0-9] [.]

<junction>              ::=      <union-junction> | <intersect-junction>
<union-junction>        ::=     ‘|’
<intersect-junction     ::=     ‘&’
<expression-list-open>  ::=     ‘(‘
<expression-list-close> ::=     ‘)’
<expression-open>       ::=     ‘(‘ | ‘{‘
<expression-close>      ::=     ‘(‘ | ‘}’
<general-delimiter>     ::=     <whitespace>
<whitespace>            ::=     ‘ ‘
<double-quote>          ::=     ‘”’
<escape-character>      ::=     ‘\’
<operator>              ::=     <equal_to> |...| <contains_or_above>

Valid operators for the preceding line:

<equal_to>              ::=     ‘=’ | ‘EQUAL_TO’
<not_equal_to>          ::=     ‘!=’ | ‘<>’ | ‘NOT_EQUAL_TO’
<in>                    ::=     ‘=’ | ‘IN’
<not_in>                ::=     ‘!=’ | ‘<>’ | ‘NOT_IN’
<greater_than>          ::=     ‘>’ | ‘GREATER_THAN’
<less_than>             ::=     ‘<‘ | ‘LESS_THAN’
```

```
<greater_than_or_equal> ::=     ‘>=’ | ‘GREATER_THAN_OR_EQUAL’
<less_than_or_equal>    ::=     ‘<=’ | ‘LESS_THAN_OR_EQUAL’
<begins_with>           ::=     ‘=*’ | ‘BEGINS_WITH’
<ends_with>             ::=     ‘*=’ | ‘ENDS_WITH’
<contains>              ::=    ‘*=*’ | ‘CONTAINS’
<not_contains>          ::=    ‘*<>*’ | ‘NOT_CONTAINS’
<in_or_below>           ::=    ‘IN_OR_BELOW’
<in_or_above>           ::=    ‘IN_OR_ABOVE’
<between>               ::=    ‘BETWEEN’
<not_between>           ::=    ‘NOT_BETWEEN’
<not_begins_with>       ::=    ‘NOT_BEGINS_WITH’
<not_ends_with>         ::=    ‘NOT_ENDS_WITH’
<is_today>              ::=    ‘IS_TODAY’
<is_not_today>          ::=    ‘IS_NOT_TODAY’
<within_last_days>      ::=    ‘WITHIN_LAST_DAYS’
<within_last_months>    ::=    ‘WITHIN_LAST_MONTHS’
<within_next_days>      ::=    ‘WITHIN_NEXT_DAYS’
<within_next_months>    ::=    ‘WITHIN_NEXT_MONTHS’
<not_within_last_days>  ::=    ‘NOT_WITHIN_LAST_DAYS’
<not_within_last_months>::=    ‘NOT_WITHIN_LAST_MONTHS’
<not_within_next_days>  ::=    ‘NOT_WITHIN_NEXT_DAYS’
<not_within_next_months>::=    ‘NOT_WITHIN_NEXT_MONTHS’
<contains_or_below>     ::=    ‘CONTAINS_OR_BELOW’
<contains_or_above>     ::=    ‘CONTAINS_OR_ABOVE’
```

Rebuilding the Apache HTTP server and PHP

This topic describes how to rebuild the Apache HTTP server and PHP and replace them in SA. SA includes an Apache HTTP server and PHP so this information is only needed if you need to use a different version of the Apache HTTP server or if you need to compile extra libraries or modules into PHP.

SA uses the Apache HTTP server and PHP for web Automation Platform Extensions (APX). For more information, see "Automation Platform Extensions (APX)" on page 96.

Extending the APX HTTP environment

This section describes how you can extend the APX HTTP environment by rebuilding the Apache HTTP server and PHP.

You must perform these tasks after all core upgrades.

If you have a Multimaster Mesh, these tasks must be performed on each slice in all cores. For more information on slice component bundles, see the Server Automation Administration Guide on the HPE SSO portal.

Rebuilding PHP

1. Perform the following tasks to rebuild PHP.

   Download the PHP source from `http://www.php.net/`.

2. Put the source in a directory on the server where `apxproxy` is installed, typically under `/opt/opsware/apxproxy`.

3. Enter the following commands, replacing the version number if you downloaded a different version of PHP.

   ```
   mkdir /build ; cp php-4.4.8.tar.gz /build; cd /build
   gzip -dc php-4.4.8.tar.gz | tar xvf -
   cd php-4.4.8
   ./configure --prefix=/opt/opsware/apxphp
   --with-pear=/opt/opsware/apxphp/lib/pear
   ```

```
--with-config-file-path=/opt/opsware/apxphp/lib
--with-apxs2=/opt/opsware/apxhttpd/bin/apxs <any other options you>
make clean
make
```

4. Backup your old copy of `libphp4.so`:

   ```
   cp /opt/opsware/apxhttpd/modules/libphp4.so
   /opt/opsware/apxhttpd/modules/libphp4.so.backup
   ```

5. Copy the new `libphp4.so` file to the `apxhhtps` directory:

   ```
   cp libs/libphp4.so /opt/opsware/apxhttpd/modules/libphp4.so
   ```

6. Ensure that the complete reference library exists in the tool.list:

   ```
   ldd ./libs/libphp4.so
   ```

   For each entry in the output ensure that the file exists in `/etc/opt/opsware/ogfs/tool.list`.

   If an entry does not exist, add it.

7. Backup the `apxphp` folder:

   ```
   mv /opt/opsware/apxphp /opt/opsware/apxphp.orig
   ```

8. Install PHP:

   ```
   make install
   ```

9. Reload and relink the OGFS to make sure anything you added to `/etc/opt/opsware/ogfs/tools.list` shows up in the OGFS:

   ```
   /opt/opsware/ogfs/tools/rewink && /opt/opsware/ogfs/
   tools/reload
   ```

10. Restart `apxproxy`:

    ```
    /etc/opt/opsware/startup/apxproxy restart
    ```

Rebuilding Apache

Perform the following tasks to rebuild the Apache HTTP server.

1. Download the source code for the Apache HTTP server from `http://httpd.apache.org/`.
2. Put the source in a directory on the server that hosts the slice component bundle. For more information on slice component bundles, see the Server Automation Administration Guide on the HPE SSO portal.
3. Enter the following commands, replacing the version number if you downloaded a different version of httpd.

   ```
   mkdir /build; cp httpd-2.2.8.tar.gz /build; cd /build
   gzip -dc httpd-2.2.8.tar.gz | tar xf -
   cd httpd-2.2.8
   ./configure --prefix=/opt/opsware/apxhttpd <any other options you want>.
   ```

   SA currently uses:

   ```
   --enable-mods-shared="actions alias auth_basic auth_digest authn_file authz_
   user cgi deflate dir dumpio env expires headers ident logio log_config mime
   negotiation rewrite userdir vhost_alias imagemap status"
   --disable-dav
   --with-port=8021
   --with-expat=builtin
   --without-pgsql
   ```

   (*On SunOS only*) Enter this command:

   ```
   perl -pi -e 's/#define HAVE_GETADDRINFO 1/#undef HAVE_GETADDRINFO/g'
   ./srclib/apr/include/arch/unix/apr_private.h
   make
   ```

4. Make a backup of the `apxhttp` directory:

   ```
   mv /opt/opsware/apxhttpd /opt/opsware/apxhttpd.orig
   ```

5. Install Apache:
   ```
   make install
   ```
6. Reload and relink the new files into the OGFS:

   ```
   /opt/opsware/ogfs/tools/rewink && /opt/opsware/ogfs/tools/reload
   ```

7. The HTTPD and the `.so` files in the modules directory may reference external libraries. These libraries must be visible (or winked in) to the OGFS.

Log in to the OGFS and run LDD on `/opt/opsware/apxhttpd/bin/httpd` and any `.so` file in `/opt/opsware/apxhttpd/modules` and ensure that all the files listed there exist in the OGFS. If they do not, add the files to `/etc/opt/opsware/ogfs/tool.list` (outside the OGFS) and then re-run step 6 until all files are available to `/opt/opsware/apxhttpd/bin/httpd.`

8. You must now rebuild PHP. See "Rebuilding PHP" on page 218.

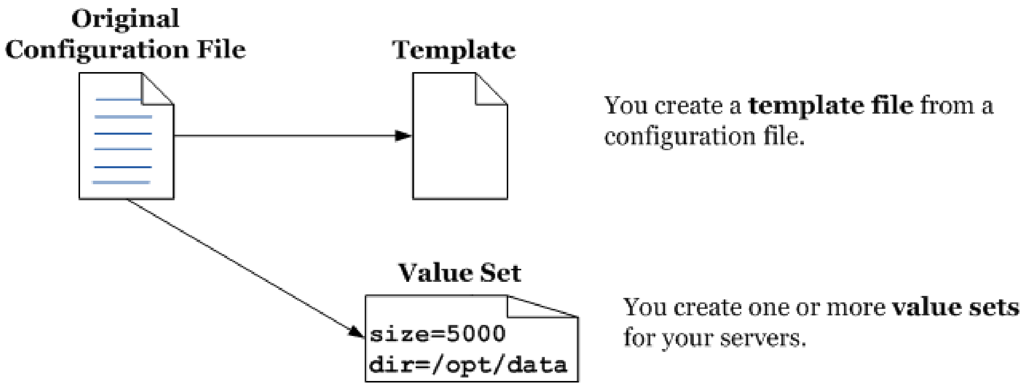
Application Configuration

This section gives you the high level view of Application Configurations. It explains the steps you need to take to set up and manage your application configurations.

- For detailed instructions on how to perform these tasks, see "Application Configuration tasks" on page 256.
- For background on Application Configurations, see "Application Configuration concepts" on page 225.

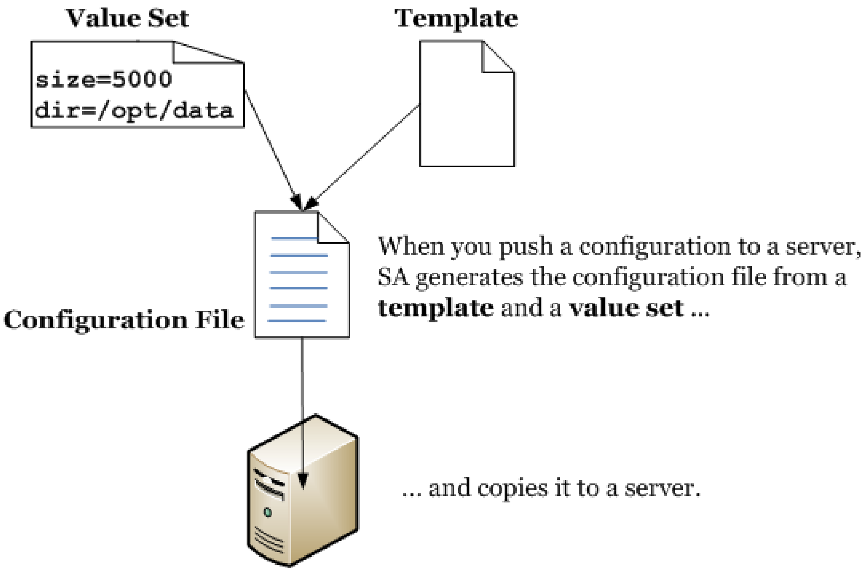
To create and use application configurations, you must first create a **template** file and a **value set**, as shown in the following figure. A template is a model of a configuration file. A value set is the data values that will be merged with the template to create an instance of the configuration file to be pushed to a server. See the following figure for detailed instructions.

Application Configuration Creation


Original Configuration File
Template
You create a **template file** from a configuration file.
Value Set
size=5000
dir=/opt/data
You create one or more **value sets** for your servers.

After creating a template and value set, you **attach** the application configuration to one or more servers and **push** the configurations to the servers, as shown in the following diagram. See the following figure for detailed instructions.

Application configuration push diagram


Value Set
size=5000
dir=/opt/data
Template
Configuration File
When you push a configuration to a server, SA generates the configuration file from a **template** and a **value set** ...
... and copies it to a server.

To create and use application configurations:

1. **Determine the configuration files you want to manage**. Choose the application or system configuration files you want to manage. For example, for the Apache Web server, you could manage the http.conf, password.conf, obj.conf, mimetypes, and magnus.conf files.
2. **Create a template file for each configuration file**. For each configuration file you want to manage, create a template file based on the configuration file. The template is a model of your original configuration file with place-holders for values that vary between servers. The template specifies what values in the configuration file are fixed and what values are variable and will be different on different servers. Also refer the following topics:
   - "Creating an Application Configuration" on page 258
   - "Creating a configuration template with Visual Editor" on page 261
   - For XML configuration files, create an XML or XML-DTD template file. For more information, see "Managing XML configuration files" on page 298.
   - For other configuration files, create a template file using the Configuration Modeling Language (CML). For more information, see "CML Reference" on page 373 and "CML Tutorial 1 - Creating an Application Configuration for a simple web app server" on page 324.
   - "Configuration templates and script templates" on page 227.
3. **Create a template file for each script**. In some cases you may need to run a script before or after updating a configuration file. You must create a template file from the scripts using CML. If you do not need to run any scripts with your configuration files, then skip this step. See also:

- "Creating a template from a script" on page 280.
- "Configuration templates and script templates" on page 227.
- "Running scripts with Application Configurations" on page 244.

4. **Import your template files into the SA Library**. Once you have created all of your templates from the configuration files and scripts, import your templates into the SA Library. Or you can create them directly in the SA Client. For more information, see "Importing and validating a template file" on page 277.
5. **Create an Application Configuration object.** An Application Configuration object is simply a container that holds one or more templates. See also:
   - "Creating an Application Configuration" on page 258.
   - "Application Configuration objects" on page 226.
6. **Add your templates to an Application Configuration object**. Once you have created and imported your templates into the SA Library, add them to an Application Configuration object. You also specify the order in which the files are installed and when the scripts, if any, are executed. For more information, see " Adding or removing templates from an Application Configuration" on page 279.
7. **Attach application configurations to servers**. Once you have created and configured your Application Configuration, attach it to the servers that use it. This creates an instance of the application configuration. You can have multiple instances of an application configuration on a server. Each instance can be for a different purpose. For example, you could have an instance that configures a staging version of the application and an instance that configures a production version of the application. Or if you had three instances of the application running on a server, you could have three instances of the application configuration to configure each instance of the application. For more information, see the following topics:
   - " Attaching an Application Configuration to a server or device group" on page 283.
   - " Application Configurations in software policies" on page 255.
8. **Create value sets for your servers**. Set the values that will be placed into the template to generate the actual configuration file that will be pushed to the server. You can set default values at multiple levels that get inherited unless they are overridden at a lower level. For more information, see the following topics:
   - "Setting values in the Value Set Editor" on page 232
   - " Value sets" on page 229
9. **Compare the actual configuration files with the configuration template**. (Optional) You can easily compare a configuration template with the actual configuration file on the server and see if there are any differences. This shows you the contents of the configuration file on the server and

the values that will be copied to the server when you push the application configuration to the server. For more information, see the following topics:

   - "Comparing a configuration template with a target configuration file - Preview" on page 293
   - "Comparing configuration templates " on page 295

10. **Push configuration changes**. To update the configuration files on servers, you “push” the application configuration to those servers. No changes are made to the actual configuration files on the server until you push those changes to the servers. Application configuration changes can be pushed to individual servers or groups of servers. For more information, see the following topics:
    - "Pushing Application Configurations" on page 287
    - "Pushing Application Configurations to servers" on page 246
11. **Audit configurations, monitor compliance and remediate**. To audit your configurations, monitor compliance and remediate configurations that have changed, see "Application Configuration compliance" on page 247 and the Server Automation Administration Guide on the HPE SSO portal.

Application Configuration concepts

With Server Automation you can manage configuration files, including XML configuration files, from a central location and easily propagate changes and updates across multiple servers in your data center. You can manage a single configuration file such as the /etc/hosts file on UNIX systems, or multiple complex configuration files associated with an application, such as the configuration files associated with a large business application such as WebLogic or Websphere.

Configuration files are any files on your servers that you need to control the contents of. This includes configuration files for applications as well as system configuration files, such as:

- Files that control the behavior of application servers, web servers, databases, or other applications. For example, you can manage the httpd.conf file for the Apache Web Server running across several different servers. You can attach an application configuration for this file and enter specific values for each instance of that file on each server.
- Files on your servers such as the /etc/hosts, /etc/fstab, /etc/passwd, or /etc/groups files on UNIX servers.
- Any other files on your servers that are used to configure your servers.

In addition, you can do the following with application configurations:

- **Run scripts** before or after the configuration file is placed on the server. For example, you could use a script that restarts the application after pushing the configurations to the server. For details, see "Running scripts with Application Configurations" on page 244.
- Use application configurations in a **software policy** as part of deployment and ongoing management of applications. For details, see " Application Configurations in software policies" on page 255.
- **Audit** to determine if servers conform to the desired application configuration file contents. For details, see "Auditing Application Configurations" on page 254.
- **Check compliance** of your servers against the application configuration you have defined. For details, see "Application Configuration compliance" on page 247.
- **Restore** a previous configuration file. For details, see "Restoring a configuration file to a previous state" on page 290.

Application Configuration objects

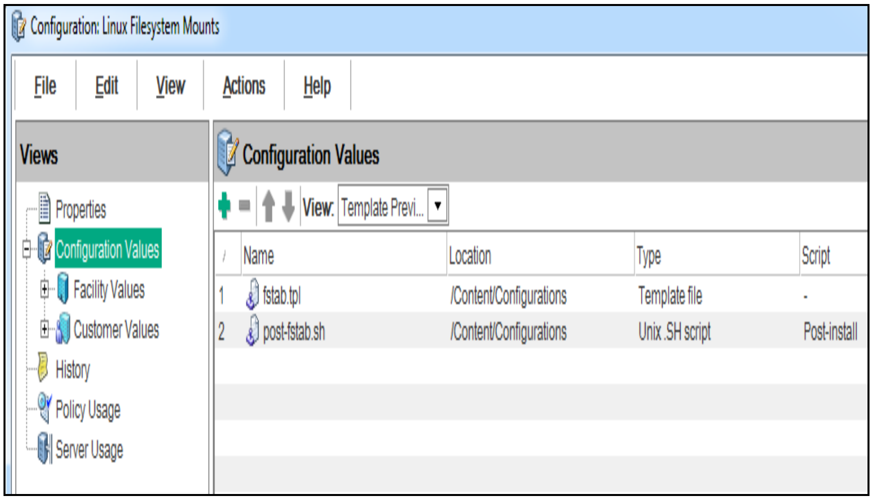
To manage a configuration file on a server, you need to create an application configuration object and at least one configuration template in the SA Library. The application configuration object is just a container for templates and optional scripts.

When you push an application configuration to one or more servers, SA does the following:

- Generates configuration files from the configuration templates and the specified value sets.
- Copies the generated configuration files to the servers.

The following figure shows an example application configuration that contains a template named “exports.tpl” and a UNIX shell script file named “post-exports.sh.” The “.tpl” file extension indicates these are templates.

Application Configuration Containing a Configuration Template and a Shell Script



View: Template Previ...

/	Name	Location	Type	Script
1	fstab.tpl	/Content/Configurations	Template file	-
2	post-fstab.sh	/Content/Configurations	Unix .SH script	Post-install

Configuration templates and script templates

A configuration template is a model of a configuration file with variables in place of the data values that vary from server to server. Configuration templates also contain instructions about how to recreate the configuration file from the template and from a value set during a push operation. The configuration template defines the fixed parts of the configuration file and the variable parts. The template also contains instructions to parse the configuration file and to recreate the configuration file to push it to managed servers.

You create configuration template files using the SA Configuration Modeling Language (CML) or in XML. Use XML for all XML-based configuration files and CML for all other configuration files. For complete information on CML, see "CML Reference" on page 373. For information on XML configuration files, see "Managing XML configuration files" on page 298.

You define the values that will be used to generate the final configuration file. These values are stored as value sets in the SA database. The CML or XML in the configuration template merges the values in a value set with the template file to generate the target configuration file. For more information, see" Value sets" on page 229.

You can also use the configuration template to import data from a configuration file and build a value set. For more information, see "Importing and validating a template file" on page 277.

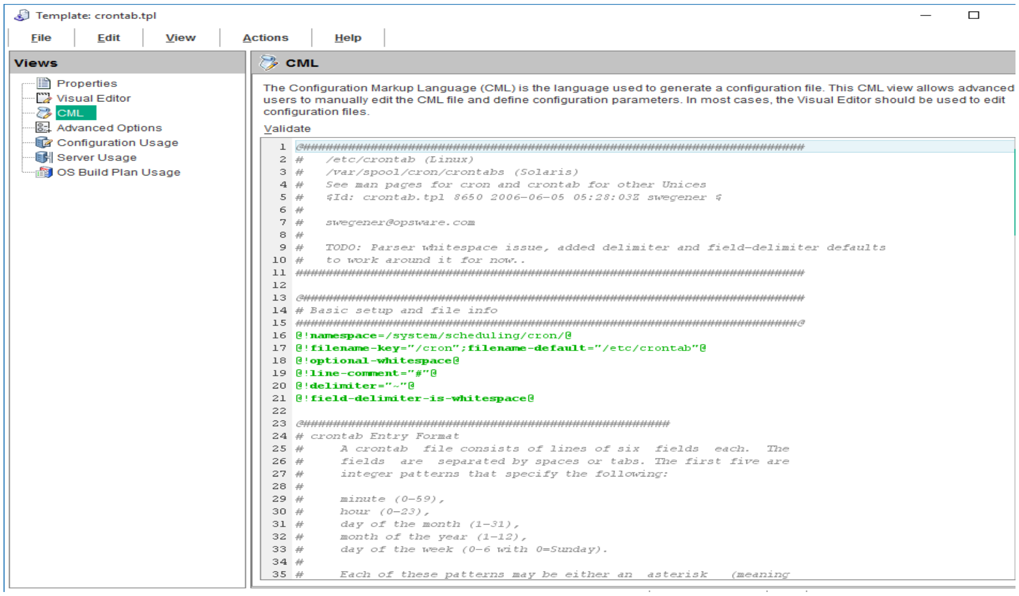
To manage configuration files, create a configuration template for each file you want to manage and add it to an Application Configuration. For more information, see "Creating an Application Configuration" on page 258.

CML configuration templates

The Configuration Modeling Language (CML) provides a way to create a template of a configuration file that is merged with a value set to generate an actual configuration file. The following figure shows a sample configuration template file in Configuration Modeling Language (CML).

For more information, see "CML Tutorial 1 - Creating an Application Configuration for a simple web app server" on page 324, "CML Tutorial 2 - Creating a template of a web server configuration file" on page 335 and "CML Reference" on page 373.

Following is an example of a configuration template written in CML.



The Configuration Markup Language (CML) is the language used to generate a configuration file. This CML view allows advanced users to manually edit the CML file and define configuration parameters. In most cases, the Visual Editor should be used to edit configuration files.

```
@##################################################################
#   /etc/crontab (Linux)
#   /var/spool/cron/crontabs (Solaris)
#   See man pages for cron and crontab for other Unices
#   $Id: crontab.tpl 8650 2006-06-05 05:28:03Z swegener $
#
#   swegener@opsware.com
#
#   TODO: Parser whitespace issue, added delimiter and field-delimiter defaults
#   to work around it for now..
###################################################################

@##################################################################
# Basic setup and file info
##################################################################@
@!namespace=/system/scheduling/cron/@
@!filename-key="/cron";filename-default="/etc/crontab"@
@!optional-whitespace@
@!line-comment="#"@
@!delimiter="~"@
@!field-delimiter-is-whitespace@

@#################################################
# crontab Entry Format
#     A crontab  file consists of lines of six  fields  each.  The
#     fields  are  separated by spaces or tabs. The first five are
#     integer patterns that specify the following:
#
#     minute (0-59),
#     hour (0-23),
#     day of the month (1-31),
#     month of the year (1-12),
#     day of the week (0-6 with 0=Sunday).
#
#     Each of these patterns may be either an  asterisk   (meaning
```

XML and XML-DTD configuration templates

You can also manage XML configuration files on your managed servers. Using XML configuration templates, you can model XML configuration file values, check those values against actual XML on target servers, and push changes to the target files. You can model XML configuration files that use a DTD as well as XML configuration files that do not.

An XML configuration template functions much like a CML template, but uses some application configuration options defined in the comments. With XML configuration templates, you can also use tags to customize the way the file is displayed in the SA Client.

For more information, see "Managing XML configuration files" on page 298.

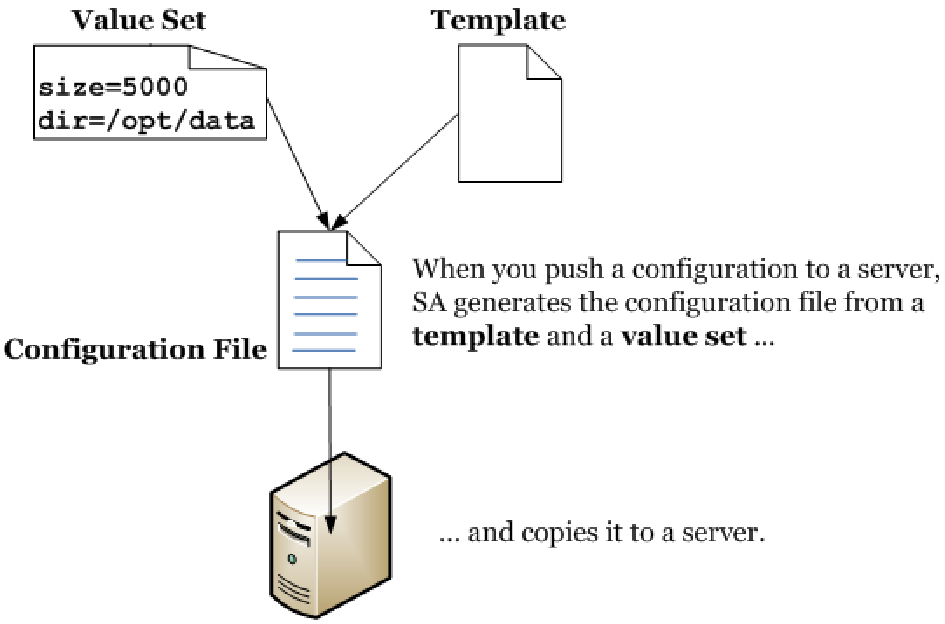
Script templates

You can add scripts to an Application Configuration that are executed before or after the configuration values are copied to the target server. To include a script in an application configuration object you must copy the script into a CML template then import the script template into the application configuration object.

For more information, see "Running scripts with Application Configurations" on page 244 and "Creating a template from a script" on page 280.

Value sets

A value set is a set of data values that are merged with a template file to generate a target configuration file. The resulting configuration file can be pushed to servers.


Value Set
size=5000
dir=/opt/data
Template
Configuration File
When you push a configuration to a server, SA generates the configuration file from a **template** and a **value set** ...
... and copies it to a server.

At its simplest, you define a value set at the “Server Instance” level for a particular server. The values in the “Server Instance” value set are merged with the configuration template to generate the actual configuration file that will be pushed to the server.

Value set levels and value set inheritance

Setting values for each individual server at the server instance level works for a small number of servers but for a large number of servers you can use value set inheritance to set default values at higher levels that apply to larger and larger subsets of your managed servers and are inherited by lower levels. Each level inherits the values from the levels above, unless the level explicitly blocks inheritance or sets a value that overrides the inherited value.

The following table lists all the value set inheritance levels, how they are either inherited or overridden at lower levels and how to set values at each level. Details on setting values at each level are described below.

Value set levels and inheritance of value sets

Level	Description of the level	How to set a value
Application	This level defines values that apply to the application configuration itself. It applies to all servers that have the application configuration attached, unless overridden by any level below.	Open the App Config object. Select Content > Application Values. Select a template. Select the “File Values” view. See "Setting values at the application level" on page 237.
Facility	This level defines values for a facility. It applies to all servers in the specified facility that have the application configuration attached, unless overridden by any level below.	Open the App Config object. Select Content > Facility Values > <*facility*>. Select a template. Select the “File Values” view. See "Setting values at the facility level" on page 238.
Customer	This level defines values for a customer. It applies to all servers belonging to the specified customer that have the application configuration attached, unless overridden by any level below.	Open the App Config object. Select Content > Customer Values > <*customer*>. Select a template. Select the “File Values” view. See "Setting values at the customer level" on page 238.
Group	This level defines values for a device group. It applies to all servers in the specified device group that have the application configuration attached, unless overridden by any level below.	Open the device group. Select Configured Applications > <*app config name*>. Select a template. See "Setting values at the group level" on page 240.
Group Instance	This level defines values for one particular instance of an application configuration. It applies to all servers in the specified device group that the instance is attached to, unless overridden by any level below.	Open the device group. Select Configured Applications > <*app config name*> > <*instance name*>. Select a template. See "Setting values at the group instance level" on page 241.

Value set levels and inheritance of value sets, continued

Level	Description of the level	How to set a value
Server	This level defines values for a server. It applies to all instances of the application configuration on the specified server, unless overridden by the level below.	Open the server. Select Management Policies tab. Select Configured Applications > <*app config name*>. Select a template. See "Setting values at the server level" on page 243.
Server Instance	This level defines values for one particular instance of an application configuration on the server. It applies only to the specified application configuration instance on the specified server and overrides all levels above.	Open the server. Select Management Policies tab. Select Configured Applications > <*app config name*> > <*instance name*>. Select a template. See "Setting values at the server instance level" on page 243.

The following table illustrates how application configuration values are inherited. The values in each row represent values set at that level. The bottom row shows the actual values that will be pushed to a server.

Table 2: Inheritance of application configuration values

Level	Values set at each level						
Application	2	1	Z	Y	X	W	V
Facility		U	T	S	R	Q	P
Customer			O	N	M	L	K
Group				J	I	H	G
Group Instance					F	E	D
Server						C	B
Server Instance							A
Inherited Results	2	U	O	J	F	C	A

Block inheritance

You can block inheritance at any level by selecting Block Inheritance in the value set editor for any variable. All values set above the blocked level will not be inherited. Values at levels below the blocked level are still inherited. A lower level must set the value or the variable will have no value.

1. Open the value set editor at any level.
2. In the value column for any variable, select a value. This displays a drop-down list and a “...” button on the right end of the edit box.
3. Select either the drop-down list or select the “...” button. This displays several choices for values of the variable.
4. Select **Block Inheritance**.
5. Select **File** > **Save** or the **Save Changes** button.

For more information, see "Setting values in the Value Set Editor" below.

Value set editor

The value set editor displays the variables defined in the template and lets you set the values of the variables. The value set editor is displayed when you select a value set of the application configuration at any of the inheritance levels. This section describes how to use the value set editor.

Setting values in the Value Set Editor

You can set any value for a variable in the value set editor as long as it conforms to the data type of the variable. Type the value in the Value column next to the desired variable.

In addition, you can set values in any of the following ways:

- Select the drop-down list on the right end of the edit box and select one of the following values:
  - Empty String - This specifies that a zero-length string is placed in the value set.
  - Block Inheritance - This specifies that values at a higher level will not be inherited. A lower level must set the value or the variable will have no value.
  - Agent version, Auth domain, Chassis ID, Customer ID, Customer Name and so forth - This specifies that the selected information about the managed server will be placed in the value set.
- Select the “...” button to the right of the edit box to display the following choices:
  - No value - Leaves the value empty in the value set.
  - Block Inheritance - This specifies that values at a higher level will not be inherited. A lower level must set the value or the variable will have no value.

  - Any Value - Enter any value.
  - Object Attribute - Select one of the values from the list above.
  - Custom Attribute - Enter a custom attribute.
  - Deployment Automation Value - Enter a value from an application managed by Deployment Automation. .
- Right click in the edit box to display the edit menu.

Setting fields in the Value Set Editor

At whichever level you are setting values, the value set editor displays the fields described below for the value set being edited.

The "Value Set Editor at the server level" on page 242 figure shows a server with an application configuration named "WAS-app-config" attached. The value set at the server instance level is displayed and can be edited. See this image for an example of the following fields.

- **Template**: Lists the template files contained in the application configuration object and lets you select the template to view and modify. Select the template you want to view.
- **Filename**: Specifies the name of the configuration file on the managed server that is the target of the configuration template. If no file name is set, then the file name is inherited. If no file name is set anywhere in the application configuration hierarchy, then the file name listed in the configuration template is used.
  If you have multiple instances of an application on a server, use this field to indicate the full path name for each target configuration file.
- **Encoding**: Specifies the character encoding for the target configuration file. The default is the encoding used on the managed server. (Note that UTF-16 encoding is not supported in the SA Client.)
- **Preserve Format**: Specifies whether to preserve spacing, comments and ordering of the target configuration file. SA will attempt to preserve as much of the target configuration file as possible, but may not be able to preserve all comments and formatting. This option is required if your template uses the `@!partial-template@` CML tag.
  If this is not turned on for the template, all comments and formatting will be removed from the file and the default ordering and spacing from the template will be used.

  > **Note:** Note that for XML-based templates, preserve format will not preserve whitespace or attribute ordering within an XML tag. Preserve format will preserve whitespace and ordering for

> everything except the whitespace in the tags themselves and the ordering of the attributes in those tags. After a push, extra whitespace inside the tags disappear and the ordering of the attributes may change. Eliminating whitespace and changing the attribute order has no effect on the meaning of the XML.

- **Preserve Values**: Specifies whether to preserve the values contained in the target configuration file on the managed server if there is no corresponding value in the value set. By default, this option is turned off.

  Preserve Values lets you request that any values in the configuration file on the server that are not also stored in the SA database are to be preserved. This is useful if you do not intend to import all the current values from the configuration file into SA, or if your users or programs sometimes modify configuration files outside of SA. This allows you to manage some configuration file changes outside of SA.
- **Show Inherited Values**: Displays the resulting value set including values inherited from higher levels. When turned off, displays only the values set at the current level. This view is read-only and is available only when viewing a value set at the server instance level.

Columns in the Value Set Editor

When you are editing value sets, the SA Client displays the following information about the configuration template values.

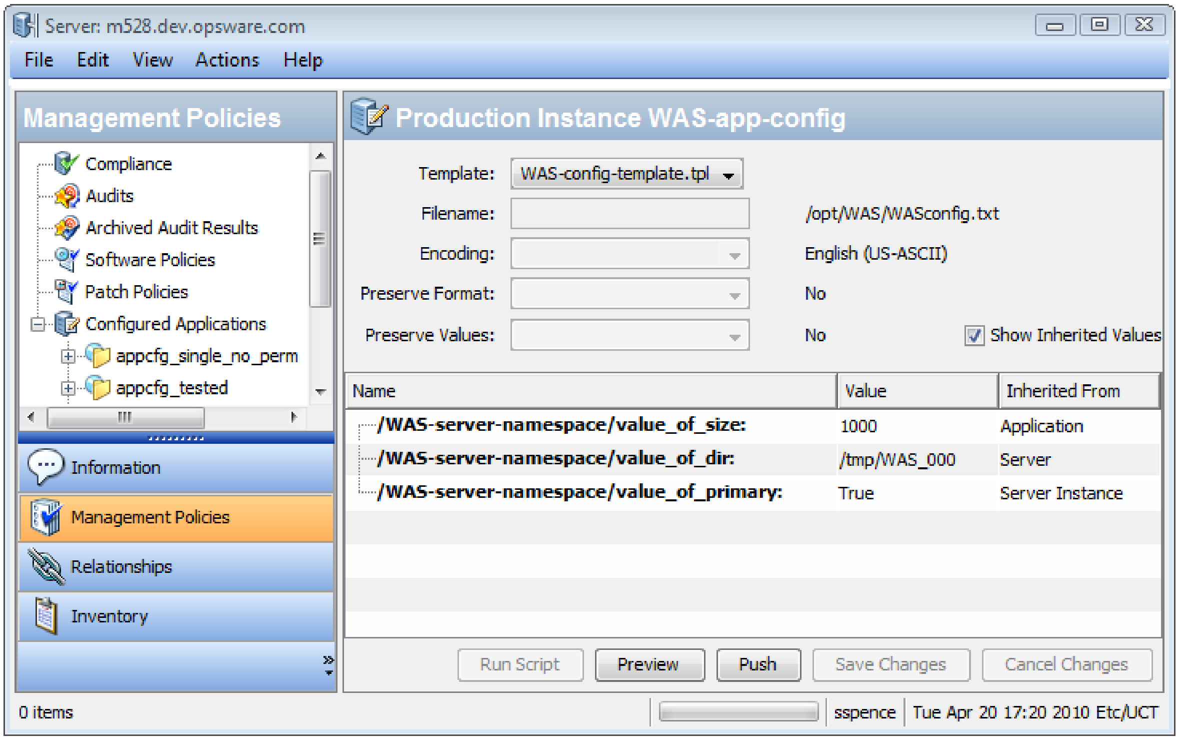
The "Value Set Editor at the server level" on page 242 figure shows a server with an instance of the application configuration named “WAS-app-config” attached. The value set at the server instance level is displayed and can be edited. The columns Name, Value and Inherited From are described below.

- **Name**: Lists the names of the variables in your template file. The name can be a simple type, a list of simple types, or a multidimensional list. Multidimensional lists are displayed beneath their parent. Elements that are required are in bold. You can double-click to show or hide multidimensional lists. To add another entry to a list type value, right-click the parent and choose **Add Item**. Required fields are set in the configuration template. Required fields cannot be empty or you will not be able to preview or push the application configuration.
- **Value**: Lists the values in value set being displayed. You can either enter a literal value or choose an attribute from the Server’s settings, such as customer name, customer ID, chassis ID, device ID, and so on. If you leave a setting blank, then the setting is inherited from its parent or ancestor (if a parent or ancestor has values configured). To set the value to a custom attribute for the value, click the “...” button to use the Set Value dialog box.

- **Inherited From**: Indicates where the value is inherited from. This column is only displayed when viewing at the server instance level and when you have Show Inherited Values selected.

  If the Preserve Values option is set, the configuration file on the server becomes the outermost level of the inheritance hierarchy. That is, if no value exists in the value set, the value in the file will be preserved.

Value Set Editor at the server instance level



Template: WAS-config-template.tpl

Filename: /opt/WAS/WASconfig.txt

Encoding: English (US-ASCII)

Preserve Format: No

Preserve Values: No

☑ Show Inherited Values

Name	Value	Inherited From
/WAS-server-namespace/value_of_size:	1000	Application
/WAS-server-namespace/value_of_dir:	/tmp/WAS_000	Server
/WAS-server-namespace/value_of_primary:	True	Server Instance

Importing value sets from an existing configuration file

While you can set values in a value set manually, you can also import values into a value set from an existing configuration file on a managed server.

To import values into a value set from an existing configuration file:

1. Display the value set editor at the level where you want to import the values. The import option is only available at the (server) instance level. See the sections below for how to display the value set editor at each level.

2. In the value set editor, make sure the Filename field shows the absolute file name of the configuration file that contains the values that you want to import.
3. Right-click in the Value column and choose **Import Values**. All of the values for the configuration template are replaced with the values from the actual configuration file.

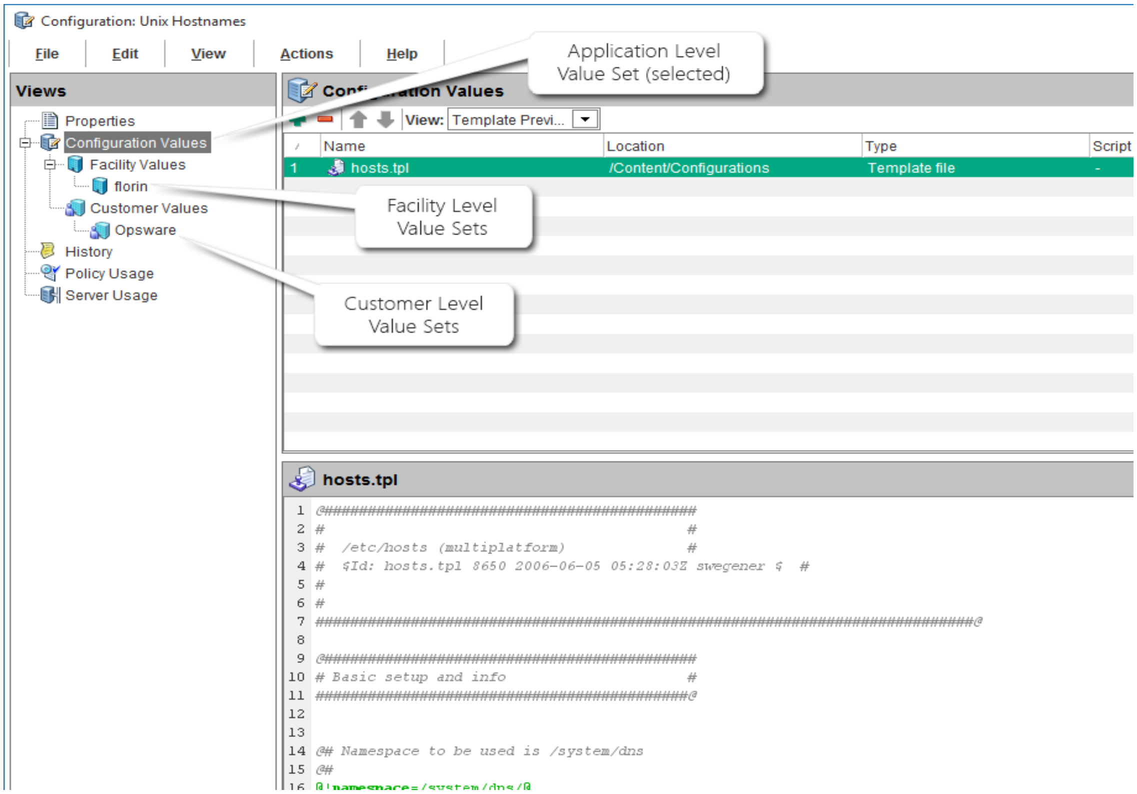
   Selecting the **Import Values** menu item reads the existing configuration file on the managed server, parses the values, and saves them at the selected level. After you import the values, you can modify any of them and push the changes back to the server, if desired
4. Select **Save Changes**.

Value Set Editor at the application, facility and customer levels

The "Value Set Editor at the application, facility and customer levels - Application level selected" below figure shows the value set editor at the application, facility and customer levels. This view is only available when you open the application configuration object.

- Selecting “Configuration Values” displays the value set editor at the application level.
- Selecting a facility under “Facility Values” displays the value set editor at the facility level.
- Selecting a customer under “Customer Values” displays the value set editor at the customer level.

Value Set Editor at the application, facility and customer levels - Application level selected



Application Level Value Set (selected)

Facility Level Value Sets

Customer Level Value Sets

/	Name	Location	Type	Script
1	hosts.tpl	/Content/Configurations	Template file	-

```
@##########################################
#                                         #
#  /etc/hosts (multiplatform)             #
#  $Id: hosts.tpl 8650 2006-06-05 05:28:03Z swegener $  #
#
#
##########################################################################@

@##########################################
# Basic setup and info                    #
##########################################@


@# Namespace to be used is /system/dns
@#
@!namespace=/system/dns/@
```

Setting values at the application level

The application level defines values that apply to the application configuration itself. It applies to all servers that have the application configuration attached, unless overridden by any level below.

To set values at the application level:

See "Value Set Editor at the application, facility and customer levels" on the previous page for an example showing the value set editor at the application, facility, and customer levels.

1. Open the application configuration object in the SA Client.
2. Select the “Configuration Values” node.
3. Select a template in the application configuration.
4. Select “File Values” from the “View:” drop-down list. This displays the value set editor in the lower right where you can set default values at the application level.
5. Enter values in the text boxes. Use the right-click menu items for additional editing capabilities.
6. Optionally select File Preview from the “View:” drop-down list to see the resulting file.
7. Select **File** > **Save to save your changes.**

Setting values at the facility level

The facility level defines values for a facility. It applies to all servers in the specified facility that have the application configuration attached, unless overridden by any level below.

To set values at the facility level:

See "Value Set Editor at the application, facility and customer levels" on page 236for an example showing the value set editor at the application, facility and customer levels.

1. Open the application configuration object in the SA Client.
2. Open the Configuration Values view to display the Facility Values node.
3. Open the Facility Values node and select a facility.
4. Select a template in the application configuration.
5. Select File Values from the View drop-down list. This displays the value set editor in the lower right.
6. Enter values in the text boxes. Use the right-click menu items for additional editing capabilities.
7. Optionally select **File Preview** from the View drop-down list to see the resulting values.
8. Select **File** > **Save** to save your changes.

Setting values at the customer level

The customer level defines values for a customer. It applies to all servers belonging to the specified customer that have the application configuration attached, unless overridden by any level below.

To set values at the customer level:

See "Value Set Editor at the application, facility and customer levels" on page 236 for an example showing the value set editor at the application, facility and customer levels.

1. Open the application configuration object in the SA Client.
2. Open the Configuration Values view to display the Customer Values node.
3. Open the Customer Values node and select a customer.
4. Select a template in the application configuration.

5. Select File Values from the View drop-down list. This displays the value set editor in the lower right.
6. Enter values in the text boxes. Use the right-click menu items for additional editing capabilities. See also "Setting values in the Value Set Editor" on page 232.
7. Optionally select File Preview from the View drop-down list to see the resulting values.
8. Select **File** > **Save** to save your changes.

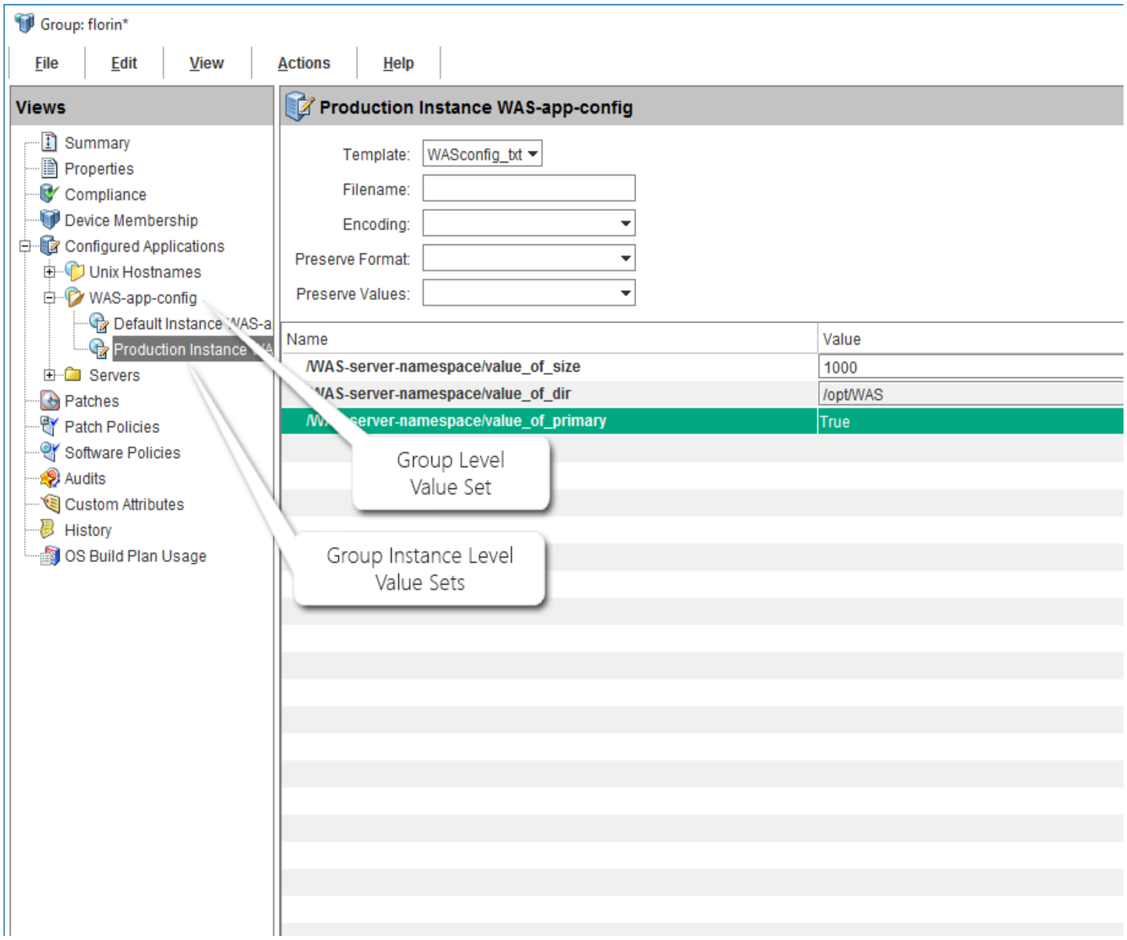
Value Set Editor at the group level

The group level defines values for a device group. It applies to all servers in the specified device group (provided the application configuration is attached to the device group), unless overridden by any level below.

"Value Set Editor at the group level and group instance level" below shows the value set editor at the group and group instance level. The group and group instance level are only available when the application configuration is attached to a device group. This view is only available when you open a device group to which the application configuration is attached. For more information, see " Attaching an Application Configuration to a server or device group" on page 283.

- Selecting the application configuration named “WAS-app-config” displays the value set editor at the group level.
- Selecting one of the application configuration instances displays the value set editor at the group instance level. This example shows two application configuration instances, named “Production Instance WAS-appconfig” and “Staging Instance WAS-appconfig”. The value set editor is displaying the value set for the “Production Instance WAS-appconfig”.

Value Set Editor at the group level and group instance level



Template: WASconfig_txt

Filename:

Encoding:

Preserve Format:

Preserve Values:

Name	Value
/WAS-server-namespace/value_of_size	1000
/WAS-server-namespace/value_of_dir	/opt/WAS
/WAS-server-namespace/value_of_primary	True

Setting values at the group level

The group level defines values for a device group. It applies to all servers in the specified device group, unless overridden by any level below.

Before you can set values at the group level, you must attach the application configuration to a device group. See the "Value Set Editor at the group level and group instance level" on the previous page figure for an example showing the value set editor at the group and group instance levels.

To set values at the group level:

1. Open the device group in the SA Client. The application configuration must be attached to the device group.
2. Open the Configured Applications node in the View pane on the left.

3. Select the desired application configuration node under the Configured Applications node. This displays the value set editor on the right.
4. Select the desired template file in the “Template:” drop-down list.
5. Enter values in the text boxes. Use the right-click menu items for additional editing capabilities.
6. Select the Save Changes button.

Setting values at the group instance level

The group instance level defines values for an instance of the application configuration attached to the device group. It applies to all servers in the specified device group, unless overridden by any level below.

Before you can set values at the group instance level, you must attach the application configuration to a device group. See the following figure for an example showing the value set editor at the group and group instance levels.

To set values at the group instance level:

1. Open the device group in the SA Client. The application configuration must be attached to the device group. See " Attaching an Application Configuration to a server or device group" on page 283.
2. Open the Configured Applications node in the View pane on the left.
3. Open the desired application configuration node under the Configured Applications node.
4. Select the desired instance of your application configuration. For example, "Value Set Editor at the group level and group instance level" on page 239 shows the application configuration instance named “Production Instance WAS-appconfig” selected. This is an instance of the application configuration object named WAS-app-config. This application configuration is attached to the device group and two instances of the application configuration are defined.
5. Select the desired template file in the “Template:” drop-down list.
6. Enter values in the text boxes. Use the right-click menu items for additional editing capabilities. See also "Setting values in the Value Set Editor" on page 232.
7. Select the Save Changes button.

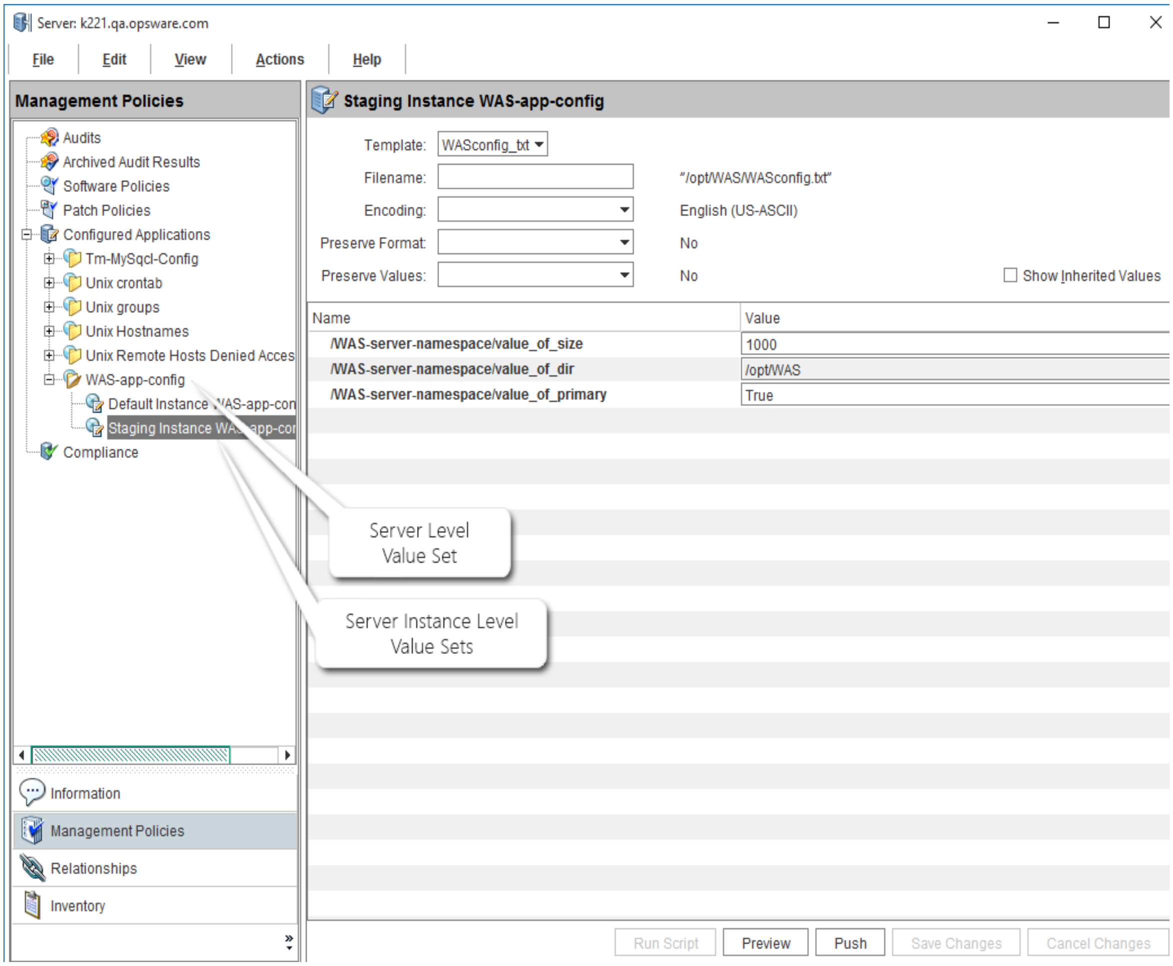
Value Set Editor at the server level

The server level defines values for a server. It applies to all instances of the application configuration on the specified server, unless overridden by the level below.

"Value Set Editor at the server level and server instance level" below shows the value set editor at the server and server instance level. This view is only available when you open a server to which the application configuration is attached.

- Selecting the application configuration named “WAS-app-config” displays the value set editor at the server level.
- Selecting either of the instances named “Production Instance WAS-appconfig” or “Staging Instance WAS-appconfig” displays the value set editor at the server instance level.

Value Set Editor at the server level and server instance level



Template: WASconfig_txt

Filename: "/opt/WAS/WASconfig.txt"

Encoding: English (US-ASCII)

Preserve Format: No

Preserve Values: No

Name	Value
/WAS-server-namespace/value_of_size	1000
/WAS-server-namespace/value_of_dir	/opt/WAS
/WAS-server-namespace/value_of_primary	True

Setting values at the server level

The server level defines values for a server. It applies to all instances of the application configuration on the specified server, unless overridden by the level below.

Before you can set values at the server level, you must attach the application configuration to a server. See "Value Set Editor at the server level and server instance level" on the previous page for an example showing the value set editor at the server and server instance levels.

To set values at the server level:

1. Open the server in the SA Client. The application configuration must be attached to the server. For details, ee " Attaching an Application Configuration to a server or device group" on page 283.
2. Select the Management Policies tab on the left.
3. Open the Configured Applications node in the Management Policies pane on the left.
4. Select the desired application configuration node under the Configured Applications node.
5. Select the desired template file in the “Template:” drop-down list.
6. Enter values in the text boxes. Use the right-click menu items for additional editing capabilities.
7. Select Save Changes.

Setting values at the server instance level

The server instance level defines values for one particular instance of an application configuration on the server. It applies only to the specified application configuration instance on the specified server and overrides all levels above.

Before you can set values at the server instance level, you must attach the application configuration to a server. See "Value Set Editor at the server level and server instance level" on the previous page for an example showing the value set editor at the server and server instance levels.

To set values at the server instance level:

1. Open the server in the SA Client. The application configuration must be attached to the server. For details, see " Attaching an Application Configuration to a server or device group" on page 283.
2. Select the **Management Policies** tab on the left.
3. Open the Configured Applications node in the Management Policies pane on the left.

4. Open the desired application configuration node under the Configured Applications node.
5. Select the desired instance under the desired application configuration node.
6. Select the desired template file in the **Template** drop-down list.
7. Enter values in the text boxes. Use the right-click menu items for additional editing capabilities.
8. Select **Save Changes**.

Running scripts with Application Configurations

You can add scripts to an Application Configuration that are executed before or after the configuration values are copied to the target server.

For example, you might want to add a **Pre-install script** that stops an application and a **Post-install script** that restarts the application after configuration changes have been made. If an error occurs during the push or post-install script, you can run a **Post-error script**.

Or you might need a **Data-manipulation script** to handle non-text configuration data. If you are configuring an IIS server, you can use a Data-manipulation script to read the metabase information into a flat file. When the information in the flat file gets parsed with the configuration template, you can run a Post-install script to write the updated information back to the metabase information.

> **Caution:** When pushing an application configuration that contains a JScript or VBScript pre-install, post-install or post-error script, the push may succeed even if the script fails. In these cases, the push ignores the script errors. The application configuration does not detect the script failure and allows the push to complete without errors. If you plan to use these types of scripts, you must make sure that the scripts are free of errors and ensure the script returns a non-zero exit status by invoking WScript.Quit(<status>).

Types of Application Configuration scripts

The following table lists the types of scripts you can use in application configuration objects. The script type specifies when the script is invoked. You can define at most one of each script type. If you define a script but do not specify one of these types, that script will be treated like a configuration template. That is, it will be pushed to the server but not executed.

Types of scripts, specifying when the script runs

Script type	Description
Data-manipulation	Runs before any pre-install script and serves the purpose of parsing a non-text configuration file to make it parseable by the CML template. The data-manipulation script is also useful when you only want to scan and import an existing file managed by the application configuration. If this script fails, the application configuration is not pushed to the server.
Pre-install	Runs before an actual push occurs. For example, a pre-install script could stop an application or service. If this script fails, the application configuration is not pushed to the server.
Post-install	Runs after the actual push occurs. For example, a post-install script could restart a service after a push.
Post-error	Runs only if the push fails or if the post-install script fails. For example, a post-error script could restore a backed-up file.

To specify the script type:

1. In the SA Client, open the application configuration object that contains the template.
2. Select the Configuration Values view to display the templates contained in the application configuration object.
3. Select the template and right-click to display the menu.
4. Select the script type, listed in the above table.
5. Select the **File** > **Save** menu.

See also "Creating a template from a script" on page 280.

The above table lists the types of scripts you can use in application configuration objects. This type specifies the syntax and execution environment of the script.

Types of script source

Script source type	Description
Windows .BAT script	Windows batch command files.
Windows .JS script	Javascript files running on Windows.
Windows .CMD script	Windows batch command files.
Windows .VBS script	Windows Visual Basic script.
Windows .WSF script	Windows script file.

Types of script source, continued

Script source type	Description
Windows .PY script	Python file running on Windows.
Unix .SH script	Unix shell scripts.
Other Unix scripts	Any other scripts that run on Unix.

To specify the script type:

1. In the SA Client, open the template.
2. Select the Properties view.
3. In the Type field, select the script type, listed in the above table.
4. Select the **File** > **Save** menu.

See also "Creating a template from a script" on page 280.

Pushing Application Configurations to servers

Anytime you change values in a value set, to merge those changes with the configuration file on the target server you must push the application configuration to the server. When you push an application configuration, all the values in the value set replace the values in the configuration files on the target managed servers. If the configuration file does not exist on the target server, a new file is created on the server when you push. In addition, all the scripts in the application configuration are executed based on the script type.

When you push an application configuration, the following events occur.

- SA runs the data-manipulation script, if specified.
- SA generates the target configuration files from the templates and value sets.
- SA backs up the existing configuration files.
- SA runs the pre-install script, if specified.
- SA copies the generated configuration files to the server.
- SA executes the post-install script, if specified.
- SA executes the post-error script, if specified and if either the pre-install script fails or the post-install script fails or the copy operation fails.

For information about how application configurations are pushed in the context of Software Policies and Audits, see " Application Configurations in software policies" on page 255.

For more information about using scripts in application configurations, see "Creating a template from a script" on page 280 and "Managing non-text configurations by running a data manipulation script" on page 281.

For instructions on how to push an application configuration, see "Pushing Application Configurations" on page 287.

> **Note:** The way in which sequences (of lists and scalars) are merged when you push depends upon how values have been set in the Application Configuration inheritance hierarchy and what sequence merge modes have been configured in the CML template for the Application Configuration. For more information about sequence merging, see "Sequence aggregation" on page 417.

Application Configuration compliance

Application configuration compliance enables you to determine whether or not the values of an application configuration attached to a server (or a group of servers) match the configuration file values on the target server.

A server is considered compliant if the target configuration file values match the values defined in the application configuration. When a target configuration does not match the values defined in the application configuration, the server is considered non-compliant.

The SA Client displays the following compliance statuses for application configurations:

- **Compliant**: All of the values in the application configurations attached to a server or device group (or several servers and groups) match the configuration values on the target server. Represented by the 🟢 icon.

  For Device Groups, AppConfig compliance is based upon the compliance status of all servers (and servers in any subgroups) that belong to a group. By default, group compliance is determined by a default threshold: if more than five percent of all servers in a group have a status of Non-Compliant, the entire group is considered Non-Compliant. To change this default setting, see the Changing Device Group Compliance Settings section in the Server Automation User Guide on the HPE SSO portal.

  - **Non-compliant**: At least one of the values defined in an application configuration does not match the values in a configuration file (or files) on a target server. Represented by the ⊗ icon.

For Device Groups, non-compliance is based upon the compliance status of all the servers (and servers in any subgroups) that belong to a group. By default, group non-compliance is determined by a default threshold: if more than five percent of all servers in a group have a status of Non-Compliant, the entire group is considered Non-Compliant.To change this default setting, see the Changing Device Group Compliance Settings section in the Server Automation User Guide on the HPE SSO portal.

- **Scan Started**: The application configuration compliance information is currently being calculated. Represented by the ⌛ icon.
- **Scan Needed**: The application configuration compliance information is undefined, perhaps because a compliance scan was never run (for example, on a new installation), or the configuration on the server (or servers in the device group) changed since the last time information was reported to the SA Client. Represented by the □ icon.
- **Not Applicable**: The application configuration compliance information does not apply and is represented by a dash (—). This is displayed if there are no application configurations attached to the server.

You can view application configuration compliance for individual servers or groups of servers:

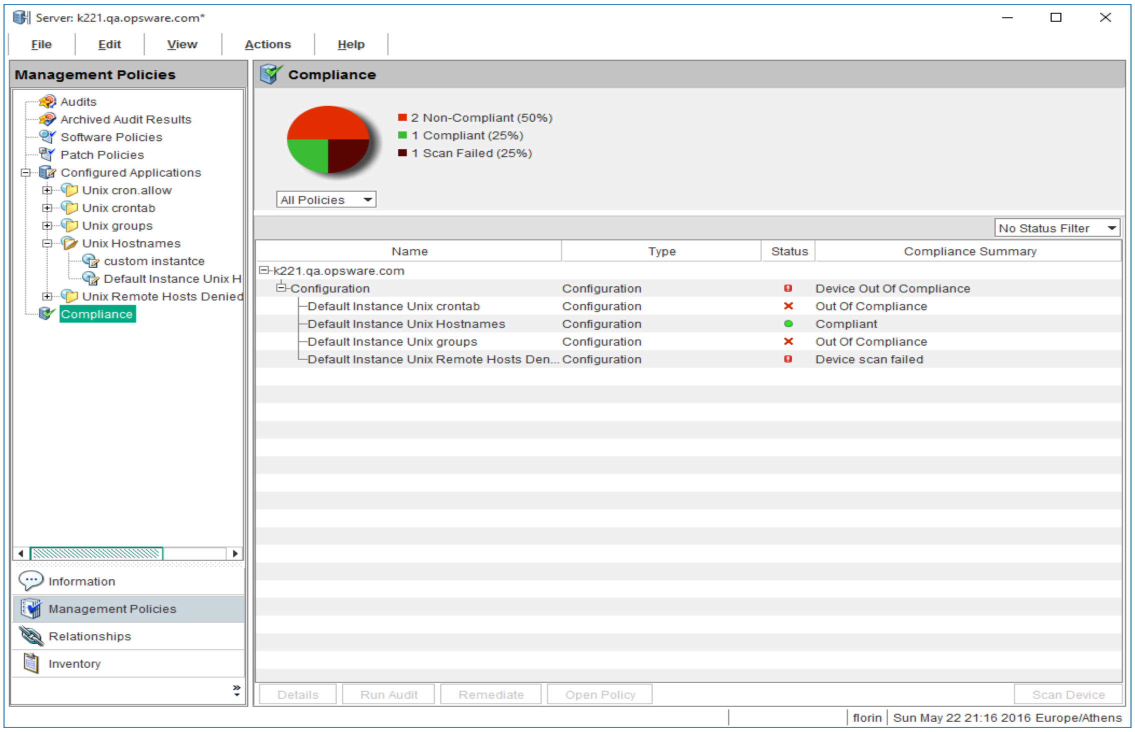
- "Application Configuration compliance for a single server" below
- "Application Configuration Compliance for multiple servers" on the next page

> **Note:** If you make any changes to an application configuration, such as editing its values in the value set editor, any server or group of servers it is attached to will cause a compliance status of Scan Needed.

Application Configuration compliance for a single server

For a single server, the compliance view displays overall compliance for all application configurations attached to the server. If more than one application configuration is attached to the server, then you can see the aggregate compliance status for all application configurations, plus each individual configuration’s compliance status.

The following figure shows a single server’s application configuration compliance.



🟥 2 Non-Compliant (50%)
🟩 1 Compliant (25%)
■ 1 Scan Failed (25%)

Name	Type	Status	Compliance Summary
⊟ k221.qa.opsware.com			
⊟ Configuration	Configuration	!	Device Out Of Compliance
Default Instance Unix crontab	Configuration	✗	Out Of Compliance
Default Instance Unix Hostnames	Configuration	●	Compliant
Default Instance Unix groups	Configuration	✗	Out Of Compliance
Default Instance Unix Remote Hosts Den...	Configuration	!	Device scan failed

If any differences are discovered between the application configuration and the actual configuration file on the target server, the lower pane shows the category that is non-compliant. If the server has several application configurations attached to it, and any one of the configuration files targeted by the application configuration is different from the application configuration, then the server’s status is non-compliant.

For more information on how to run an application configuration compliance scan, see "Scanning servers for Application Configuration compliance" on page 253.

Application Configuration Compliance for multiple servers

You can view the application configuration compliance status for multiple servers. From the SA Client navigation pane, select Devices then select Device Groups or Servers. Select a device group or a set of servers, then from the View menu select Compliance. This displays the aggregate compliance status for the selected servers.

An application configuration attached to a group of servers is considered compliant if less than five percent of the servers in the group are out of compliance. If over five percent are out of compliance, the aggregate compliance is considered non-compliant. You can change this percent by selecting the Administration tab in the SA Client, then selecting Compliance Settings.

The details pane for a group of servers in the Compliance view shows whether or not all of the application configurations are compliant, but does not expand to show a breakdown of individual servers and application configurations.

You can view server group application configuration compliance status in the following ways:

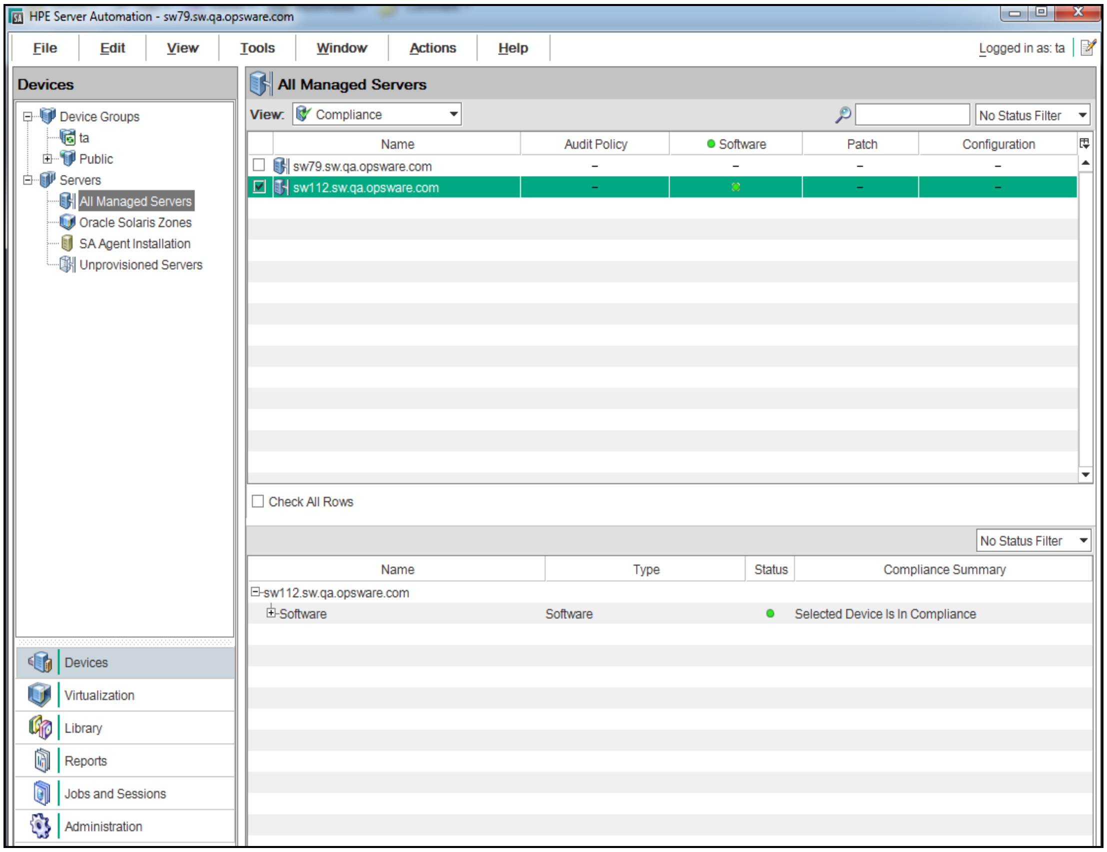
- "Viewing Application Configuration compliance for multiple servers" below.
- "Viewing Application Configuration compliance for multiple device groups" on the next page.
- "Viewing Application Configuration compliance for a single devicegroup" on page 252.

Viewing Application Configuration compliance for multiple servers

To view application configuration compliance for multiple servers:

1. From the SA Client Navigation pane, select **Devices** > **Servers** > **All Managed Servers**.
2. From the View drop-down list, select **Compliance**.
3. To see compliance levels for more than one server, select the check box next to the servers, and a roll up of compliance for the selected servers displays in the bottom details pane, as shown in the

following figure.



View: Compliance

No Status Filter

	Name	Audit Policy	● Software	Patch	Configuration
□	sw79.sw.qa.opsware.com	–	–	–	–
☑	sw112.sw.qa.opsware.com	–	✓	–	–

□ Check All Rows

No Status Filter

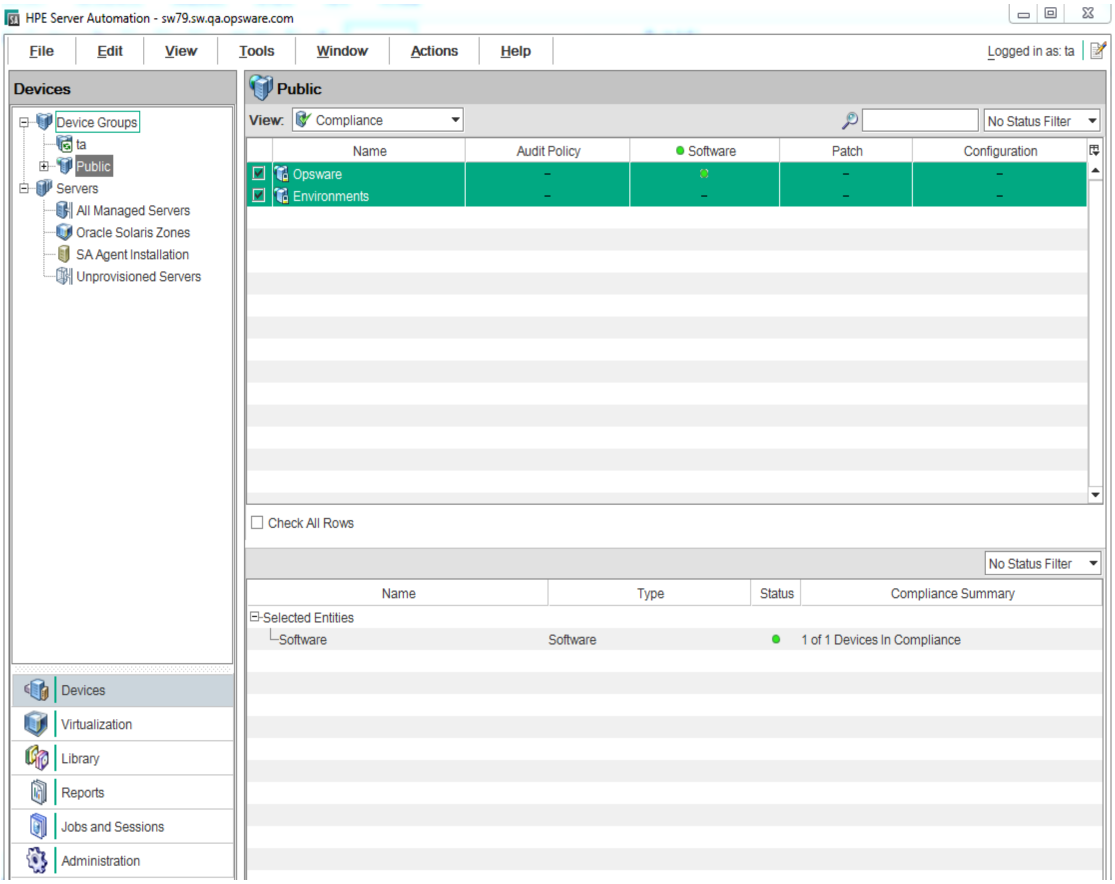
Name	Type	Status	Compliance Summary
⊟-sw112.sw.qa.opsware.com			
⊞-Software	Software	●	Selected Device Is In Compliance

Viewing Application Configuration compliance for multiple device groups

To view application configuration compliance for multiple device groups:

1. From the SA Client navigation pane, select **Devices** > **Device Groups**.
2. Select a device group or a folder containing device groups.
3. From the View drop-down list, select Compliance. This displays the compliance status for all the groups.
4. To see compliance levels for more than one group, select the check box next to the servers, and a summary of compliance for the selected groups displays in the bottom details pane, as shown in

the following figure.



View: Compliance

	Name	Audit Policy	● Software	Patch	Configuration
☑	Opsware	–	●	–	–
☑	Environments	–	–	–	–

□ Check All Rows

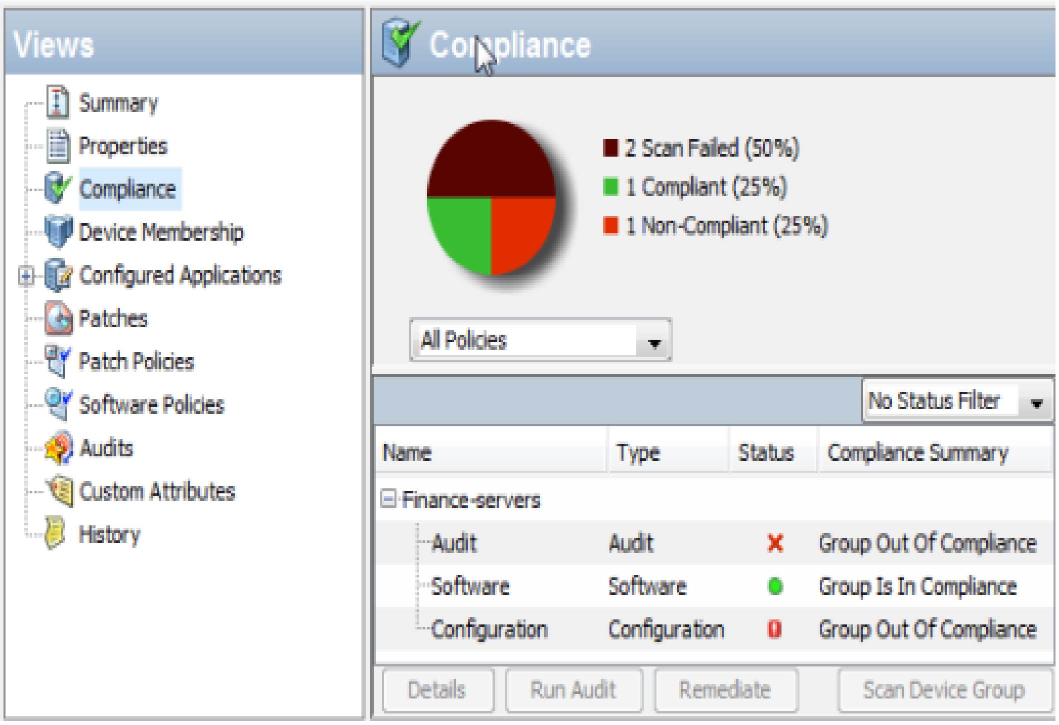
Name	Type	Status	Compliance Summary
⊟Selected Entities			
└Software	Software	●	1 of 1 Devices In Compliance

Viewing Application Configuration compliance for a single devicegroup

To view application configuration compliance for one device group:

1. From the SA Client navigation pane, select **Devices** > **Device Groups**.
2. Navigate to the desired device group and select it.
3. Right-click and select **Open** or select **Actions** > **Open**. This displays the device group.
4. From the Views pane, select Compliance. This displays aggregate compliance for each policy type for all members of the group as a whole, as opposed to compliance status for each individual server, as shown in the following figure.

Application Configuration compliance for a device group



- 🟥 2 Scan Failed (50%)
- 🟩 1 Compliant (25%)
- 🟥 1 Non-Compliant (25%)

Name	Type	Status	Compliance Summary
Finance-servers			
Audit	Audit	✗	Group Out Of Compliance
Software	Software	●	Group Is In Compliance
Configuration	Configuration	0	Group Out Of Compliance

Scanning servers for Application Configuration compliance

After an application configuration has been pushed to a server, the configuration file on the server can be changed or altered, either intentionally or by accident. Or the values defined in the application configuration may have changed. When a configuration file’s values on a target server do not match the values defined in the application configuration, the configuration file is considered non-compliant.

You can scan for configuration compliance on a server to determine if any of the configuration files on the server are out of compliance with the values stored in the configuration templates. You can schedule the scan to occur at regular intervals.

To scan a server or multiple servers for configuration compliance:

1. From the SA Client Navigation pane, select Devices.
2. Select either Device Groups or All Managed Servers. If you selected Device Groups, select a device group to display the servers that belong to it.

3. From the content pane, select a server. You can also select multiple servers or device groups and scan them all.
4. From the **Actions** menu, select **Scan > Configuration Compliance** or select **Schedule** > **Configuration Compliance Scan.**
   - If you selected **Scan > Configuration Compliance,** SA scans the devices to determine compliance and displays the status in the Scan Configuration Compliance screen.
   - If you selected **Schedule > Configuration Compliance Scan**, SA displays the Schedule Job screen where you can specify when you want the job to complete and other job parameters.

Auditing Application Configurations

With SA you can audit configuration files on servers to determine whether or not those files meet your organization's configuration standards. You can create audit rules that specify how a configuration file on your servers should be defined and audit those servers regularly to check that a configuration file is configured properly. If you find a mismatch between the audit rule definition and the target configuration file values, you can remediate the servers to fix the problem.

For example, to ensure that an /etc/hosts file on a managed server only defines certain host names for a specific IP address, you can define an audit rule that specifies the acceptable list of host name and IP address pairs. When you run the audit, if the hosts file contains any values other than what you specified in the rule, the audit results will show an error and you can remediate the problem.

The general process of auditing application configurations follows these steps:

1. **Create an Audit and Audit Rule**: To audit a configuration file on a server, you first create an audit. When you create the audit, you specify a source server (or a snapshot or a snapshot specification) upon which the configuration rule will be based. Then you select an application configuration template to construct the rule. The rule defines the exact values you want to check in the target configuration files. For each audit rule, specify the location of the configuration file on the target server.
2. **Select Target Servers**: In the audit, select the target servers for the audit. You can select a single server, multiple servers, or groups of servers.
3. **Run or Schedule the Audit**: You can schedule the audit to run once or on a recurring basis. You can also specify email addresses where audit results will be sent.
4. **Check Audit Results**: Check the audit results to see if the configuration files on the target servers match the values defined in the audit rule. If there are discrepancies, you can compare the rule and the target file to see the differences so you can decide how to remediate the servers.

5. **Remediate Servers**: To fix any differences found in the audit results, you can remediate the servers or any of the rules or all of the rules, to ensure that the target configuration matches the rule.

For more information on using audits and snapshots, see the "Audit and Remediation" section in the the SA 10.50 Administration Guide.

Application Configurations in software policies

Application configurations can be a powerful tool when used inside a software policy. A software policy defines an ideal state of an application including all the packages, patches, scripts, and other objects to be installed on a server, as well as the way configuration files for the application should be set on the server. When you install the software policy on a managed server, SA applies all the contents to the servers targeted by the policy, including all the values defined in the application configuration.

Using the compliance view in the SA Client, you can view the compliance status of the software installed from the policy. For example, if someone removes a patch from the software policy or installs a new package on the server or changes one of the configuration files defined in the policy, the policy will show as out of compliance in the compliance view. To make sure the application is installed and configured correctly, you can remediate the server.

For more information on using and creating software policies, see "Software Management" section in the the SA 10.50 User Guide.

To use application configurations in a software policy:

1. **Define Application**:Before building a software policy, an application expert gathers all the necessary packages and patches that comprise the application. In addition, gather the configuration templates that define and manage the configuration files associated with the application.
2. **Import Packages and Patches into SA**: Once the components of a software policy have been defined, import all the packages and patches that comprise the application into the SA Library, so they can be placed in the software policy.
3. **Create Application Configuration and Set Values**: Define the configuration values that will be used to generate the configuration files. For example, if the software policy is being created to deploy an Apache Web Server, the application expert uses the value set editor to define the default values for the httpd.conf file. Add any pre- or post-installation scripts to the application

configuration, for example, to restart the Apache service after the application configuration is pushed during the software policy remediation.

4. **Test Application Configurations**: Before adding the application configurations to a software policy and deploying the application to a server, it is a good idea to attach the application configuration to a server and make sure that the application is working properly before creating the software policy. You can preview pushing configurations to a server to verify their correctness.
5. **Create Software Policy**: Once all of the components of the software policy have been defined, created, and imported into SA, the application expert creates a software policy that specifies the software to be installed, the order in which its components will be installed, including all of the patches, packages, and application configurations. When the software policy is saved in the SA Library, it is then accessible to the systems administrators who deploy, test, and manage the application.
6. **Attach Policy to Servers or Groups of Servers**: After the software policy has been created and saved, the system administrator attaches the policy to a server or group of servers in a device group.
7. **Remediate Servers to Install the Software**: The system administrator deploys the software to one or more servers by remediating the software policy on servers. Remediation ensures that everything defined in the policy is deployed on the target servers in the order specified in the policy. Prior to remediation, the administrator can preview application configurations in the job. The preview step provides the option to define the appconfig values for the template variables at the server-instance level.
8. **Test Application and Iterate Changes**: After the system administrator installs the application using software policy remediation, before the application is put into a production environment, the application should be tested to make sure it works properly and contains the correct components. In addition, each part of the application that is affected by its configuration files should be checked to ensure it is configured properly.
9. **Roll Out the Application**: After the application is deployed and in use, the system administrator can perform ongoing management and maintenance tasks, such as running software compliance scans to determine the compliance status of servers where the application is deployed, remediating non-compliant servers, and generating software compliance reports.

For more information on using software policies, see the SA 10.50 User Guide.

Application Configuration tasks

This section describes how to perform application configuration tasks.

Getting started with application configurations and templates:

- "Creating an Application Configuration" on the next page
- "Creating a configuration template" on page 259
- "Creating a configuration template with Visual Editor" on page 261

Editing and managing application configuration templates:

- "Editing a configuration template with Visual Editor" on page 268
- "Editing CML or XML in a template" on page 276
- "Importing and validating a template file" on page 277
- "Viewing configuration template sources" on page 278
- " Adding or removing templates from an Application Configuration" on page 279
- " Specifying the template order in the Application Configuration" on page 279

Using scripts in application configurations:

- "Creating a template from a script" on page 280
- "Managing non-text configurations by running a data manipulation script" on page 281
- "Running data manipulation scripts manually" on page 282

Attaching and detaching application configurations:

- " Attaching an Application Configuration to a server or device group" on page 283
- "Detaching an Application Configuration from a server or device group" on page 285

Pushing and managing application configurations:

- "Pushing Application Configurations" on page 287
- " Scheduling an Application Configuration push" on page 290
- "Restoring a configuration file to a previous state" on page 290
- "Searching and filter job results" on page 293
- "Comparing a configuration template with a target configuration file - Preview" on page 293
- "Comparing configuration templates " on page 295
- "Localizing configuration file element names" on page 296

Creating an Application Configuration

An application configuration is a container for configuration template files and, optionally, scripts that are executed when the application configuration is pushed to servers. For details, see "Application Configuration objects" on page 226.

To create an application configuration:

1. From the SA Client navigation pane, select **Library** and then select the **By Type** tab.
2. Locate and open the Application Configuration node:
   - Open the Configurations node.
   - Open an operating system group that is relevant to the application configuration.
   - Select an operating system.

   > **Note:**
   > an Application Configuration can apply to multiple operating systems. You can specify this in a later step.
3. From the menu, select **Actions** > **New** to open a new Configuration window where you can specify the configuration properties and contents.
4. In the Properties view, specify the name and description of the configuration. In addition, specify the following:
   - **Location: Specify where in the SA Library you want to store the Application Configuration.**
   - **Version**: The version can be any string you use for tracking changes to the Application Configuration. The version is not incremented automatically.
   - **OS**: Specify to which operating systems this application configuration applies.
     - Only servers with the specified operating systems will be able to use the Application Configuration.
     - Only templates that are applicable to all the operating systems defined here can be contained in this application configuration; that is, the OSs for the configuration must be a subset of those for the templates.

- **Availability**: Use this setting to keep track of which Application Configurations are tested and ready to be used and which are not yet tested or are deprecated. This setting does not change what can be done with the Application Configuration.
- **Localization Files**: Select the “+” button to add templates for generating configuration files in local languages. For details, see "Localizing configuration file element names" on page 296.

5. In the Configuration Values view, select the **Actions > Add** menu or the “+” button to add a template to the Application Configuration.
   - Use the “-” button to remove the selected configuration templates.
   - Use the up and down arrows to change the order in which the configuration templates are installed.
   - Select the Template Preview view to see the contents of the selected template file.
   - Select the File Values view to see what values will be placed into the selected template file to generate the configuration file.
   - Select the File Preview view to see what the selected configuration file will look like with the current value set.
6. Select **File > Save** to save your Application Configuration.

Creating a configuration template

A configuration template is similar to a native application configuration file, but with its variable parts “templatized” with the Configuration Modeling Language (CML) and instructions for moving values between the configuration file and the value set. See "Configuration templates and script templates" on page 227.

Any scripts that you want executed with the application configuration must also be written in CML template format. For more information, see "Creating a template from a script" on page 280.

To create a configuration template:

1. From the SA Client navigation pane, select Library and then select the By Type tab.
2. Locate and open the Application Configuration node. Open the Templates node. Open the operating system group and select the operating system where the template file will be used. Note that a template can apply to multiple operating systems. You can specify this in a later step.
3. Select the **Actions** > **New** menu. This displays the Templates screen where you define the properties of the template.

4. Select the Properties view and enter the name and description of the template file as well as the following information:
    - **Location:** Specify where in the **SA** Library you want to store the template.
    - **Version**: The version can be any string you use for tracking changes to the template. The version is not incremented automatically.
    - **Type**: Specify whether it is a template file, a script or a localization file:
        - A template file is a model of configuration file.
        - Scripts are executed before or after pushing configuration files to the server. For more information, see "Creating a template from a script" on page 280.
        - A localization file is used for configuration files customized for different locales. For more information, see "Localizing configuration file element names" on page 296.
    - **Parser Syntax**: Select the type of syntax used by the template:
        - CML Syntax for all text configuration files other than XML files, and for all script files.
        - XML Syntax for configuration files written in XML.
        - XML DTD Syntax for configuration files written in XML that also use a DTD.
        - Custom Attribute Syntax for OS Provisioning
    - **OS**: Specify to which operating systems this application configuration template applies.
        - Only servers with the specified operating systems will be able to use the template.
        - This template can only be used by Application Configurations that are applicable to one or more of the operating systems listed here.
    - **Availability**: Use this setting to keep track of which templates are tested and ready to be used and which are not yet tested or are deprecated. This setting does not change what can be done with the template. You can use this field value as a search criteria.
    - **Auditable**: Set this when you want to enable auditing for this template file. For more information, see "Auditing Application Configurations" on page 254.
5. Select the Content view.
6. Enter your CML or XML or XML DTD text directly in the template editor. See "Editing CML or XML in a template" on page 276 for the editing operations and syntax highlighting. For details on CML and XML, see the "CML Reference" on page 373 and "Managing XML configuration files" on page 298.
7. Select **Validate** to parse the CML or XML syntax and check for errors.
8. Select **File** > **Save** to save your template.

9. Add your template to an application configuration object. See " Adding or removing templates from an Application Configuration" on page 279.

Creating a configuration template with Visual Editor

The Visual Editor mode enables you to create a simple application template without having to learn CML or understand the whole configuration file. This is a useful tool if you want to parameterize a small section of a configuration file to push changes to a server.

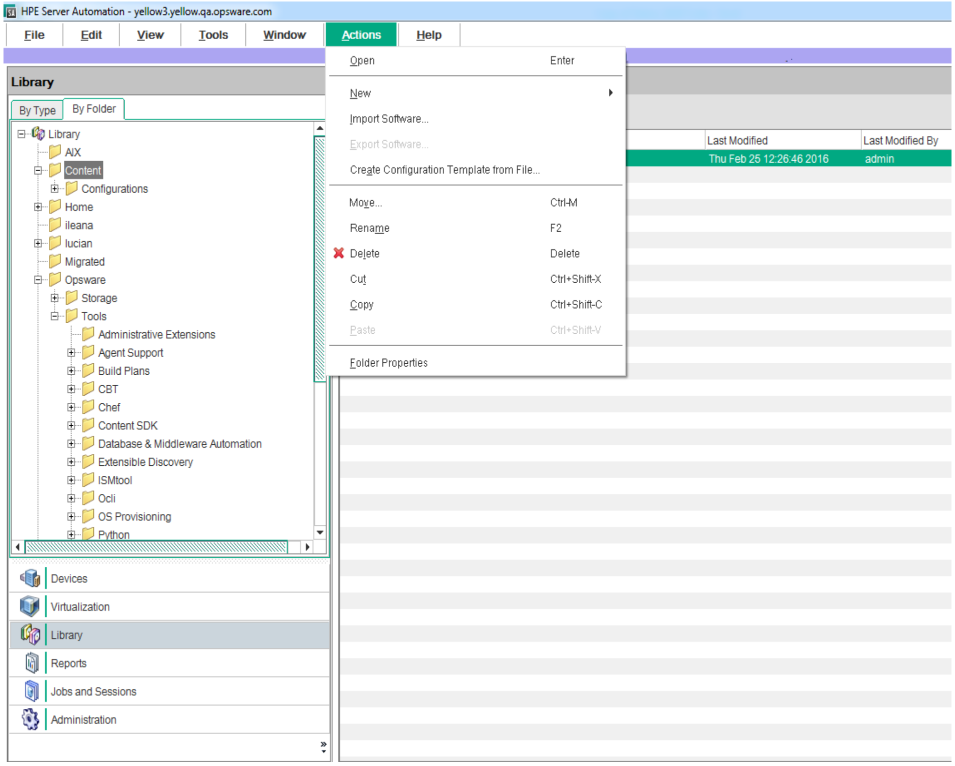
1. To begin, you can create an application configuration template using the Visual Editor by starting with an existing file, such as:
   - A working configuration file on an SA managed server
     See "Creating a template from a working configuration file" on the next page.
   - A sample configuration file imported from the local PC
     See "Importing and creating a template" on the next page.
2. Then, the Visual Editor mode provides a user interface for editing a simple application configuration template without using CML.
   - Simply mark the configuration options that should be replaced at configuration push time and the form view appears. The marked options populate the parameter name, display name, and data type fields based on the underlying template. The form view allows you to modify the parameter details as desired. See "Editing a configuration template with Visual Editor" on page 268.
3. After creating the template, you have the option of instantly generating the application configuration instance.
   - From a working configuration file on a managed server, you can view a list of compatible templates, open a template, and instantly generate the application configuration instance and attach it to the server. The configuration instance opens in the value set editor, where you can modify the values and push the changes to the server. See "Managing configuration files" on page 272

> **Note:** The ability to perform specific actions in SA is governed by your permission settings. Additionally, in order to access the Visual Editor through the server file system, you need to have OGFS permission to read the file system. To obtain additional permissions, contact your SA Administrator. See the SA 10.5 Administration Guide for more information.

Importing and creating a template

1. From the SA Client navigation pane, access the folder where you wish to import the configuration file from your PC by selecting **Library** > **By Folder** or > **By Type** and navigate to the desired folder.

2. From the **Actions** menu, select **Create Configuration Template from File...**.



3. Find and select the configuration file you wish to import and click **Open**.

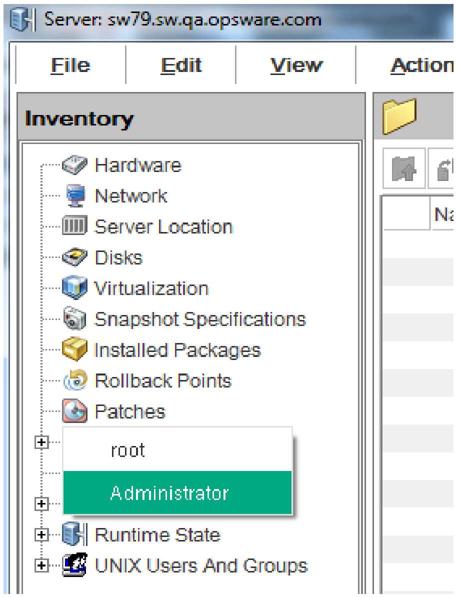
4. A template of the selected file opens in Visual Editor mode.

   See "The Visual Editor interface" on page 264 and "Editing a configuration template with Visual Editor" on page 268.

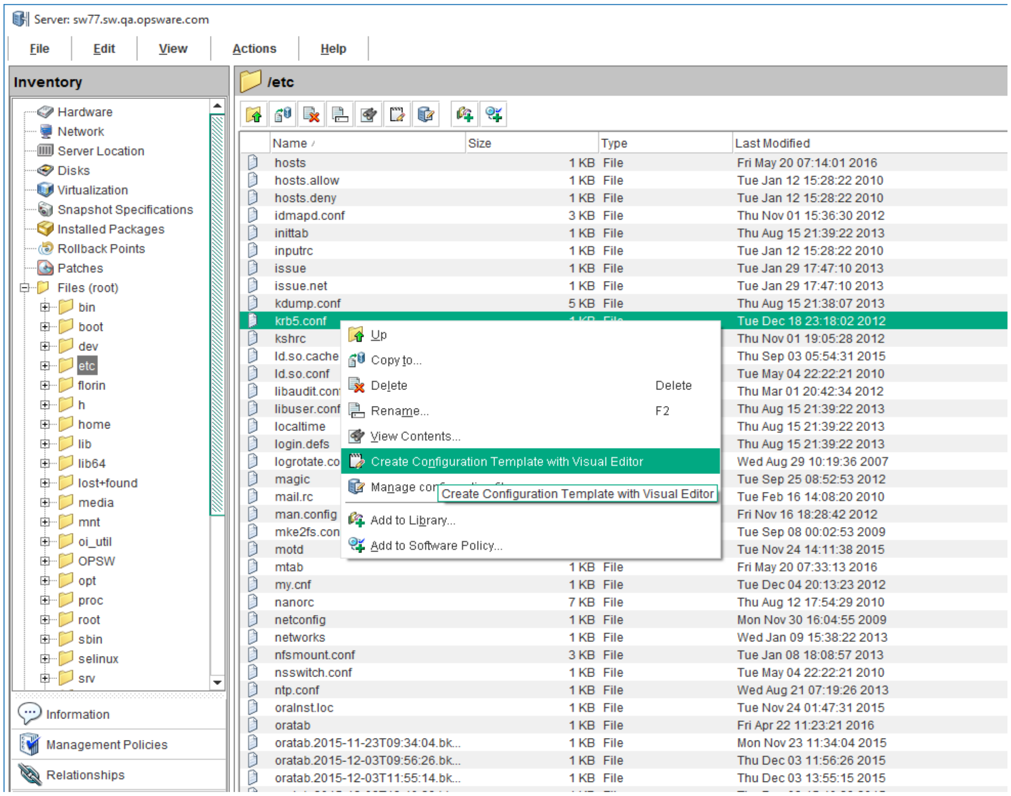
Creating a template from a working configuration file

To create a configuration template with the Visual Editor from a working configuration file:

1. Open a server browser:
   a. From the SA Client navigation pane, access the list of managed servers or device groups:
      i. Select **Devices** > **Servers** > **All Managed Servers** to view the server list.
      ii. Select **Devices** > **Device Groups** to view the device group list.
   b. From the content pane, select the server or device group you want to open.
   c. From the **Actions** menu, select **Open**.
2. Navigate to **Server Browser** > **Inventory** > **Files**.
3. When prompted, choose the appropriate root path for the server file system (typically, root for UNIX or Administrator for Windows).



4. Right-click the configuration file and choose **Create Configuration Template with Visual Editor**.



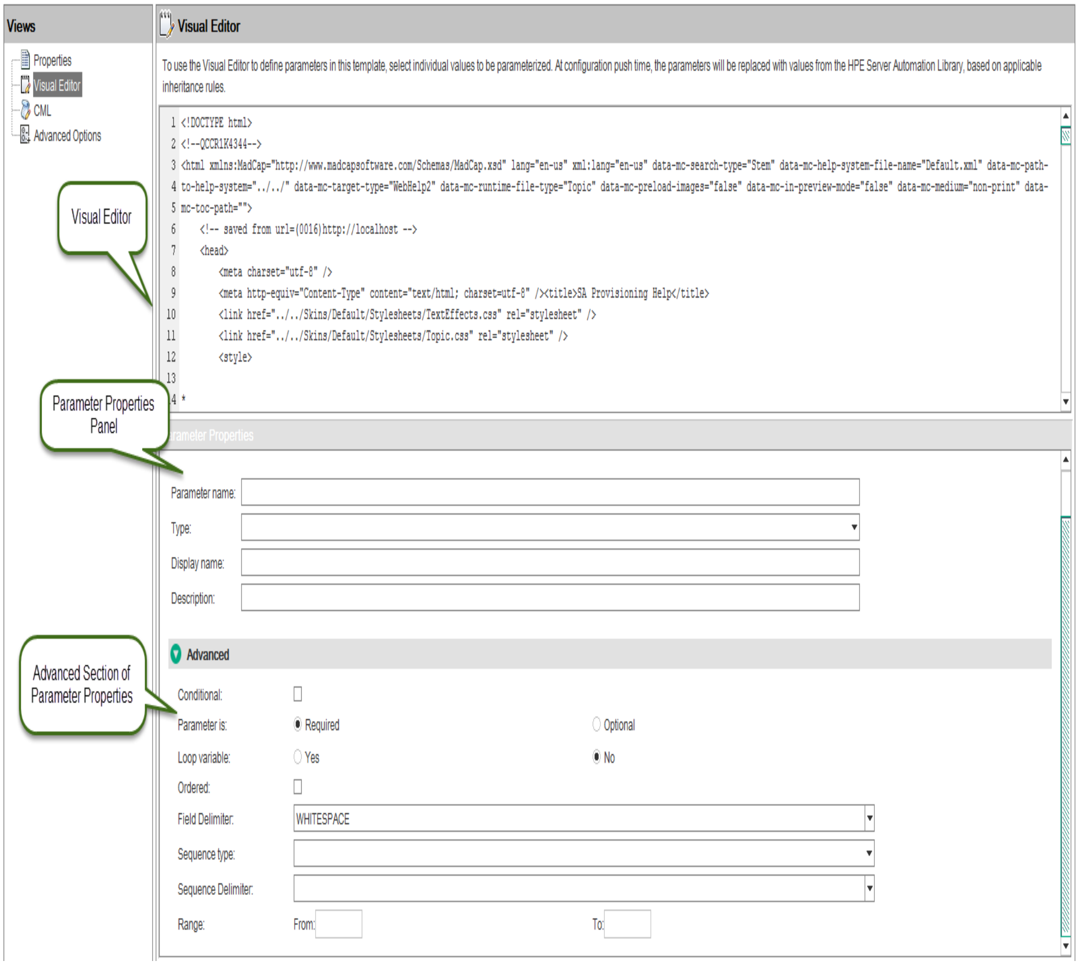
Name	Size	Type	Last Modified
hosts	1 KB	File	Fri May 20 07:14:01 2016
hosts.allow	1 KB	File	Tue Jan 12 15:28:22 2010
hosts.deny	1 KB	File	Tue Jan 12 15:28:22 2010
idmapd.conf	3 KB	File	Thu Nov 01 15:36:30 2012
inittab	1 KB	File	Thu Aug 15 21:39:22 2013
inputrc	1 KB	File	Tue Jan 12 15:28:22 2010
issue	1 KB	File	Tue Jan 29 17:47:10 2013
issue.net	1 KB	File	Tue Jan 29 17:47:10 2013
kdump.conf	5 KB	File	Thu Aug 15 21:38:07 2013
krb5.conf	1 KB	File	Tue Dec 18 23:18:02 2012
kshrc			Thu Nov 01 19:05:28 2012
ld.so.cache			Thu Sep 03 05:54:31 2015
ld.so.conf			Tue May 04 22:22:21 2010
libaudit.con			Thu Mar 01 20:42:34 2012
libuser.conf			Thu Aug 15 21:39:22 2013
localtime			Thu Aug 15 21:39:22 2013
login.defs			Thu Aug 15 21:39:22 2013
logrotate.co			Wed Aug 29 10:19:36 2007
magic			Tue Sep 25 08:52:53 2012
mail.rc			Tue Feb 16 14:08:20 2010
man.config			Fri Nov 16 18:28:42 2012
mke2fs.con			Tue Sep 08 00:02:53 2009
motd			Tue Nov 24 14:11:38 2015
mtab	1 KB	File	Fri May 20 07:33:13 2016
my.cnf	1 KB	File	Tue Dec 04 20:13:23 2012
nanorc	7 KB	File	Thu Aug 12 17:54:29 2010
netconfig	1 KB	File	Mon Nov 30 16:04:55 2009
networks	1 KB	File	Wed Jan 09 15:38:22 2013
nfsmount.conf	3 KB	File	Tue Jan 08 18:08:57 2013
nsswitch.conf	1 KB	File	Tue May 04 22:22:21 2010
ntp.conf	1 KB	File	Wed Aug 21 07:19:26 2013
oralnst.loc	1 KB	File	Tue Nov 24 01:47:31 2015
oratab	1 KB	File	Fri Apr 22 11:23:21 2016
oratab.2015-11-23T09:34:04.bk...	1 KB	File	Mon Nov 23 11:34:04 2015
oratab.2015-12-03T09:56:26.bk...	1 KB	File	Thu Dec 03 11:56:26 2015
oratab.2015-12-03T11:55:14.bk...	1 KB	File	Thu Dec 03 13:55:15 2015

A template of the selected configuration file opens in Visual Editor mode. See "The Visual Editor interface" below.

The Visual Editor interface

The Visual Editor interface displays the underlying application configuration template details in a split panel, where the lower panel displays the detailed properties of the selected parameter in an editable form view. The Parameter Properties panel appears when you first enter the Visual Editor window.

Visual Editor interface



Views
Properties
Visual Editor
CML
Advanced Options
Visual Editor
Parameter Properties Panel
Advanced Section of Parameter Properties

Visual Editor

To use the Visual Editor to define parameters in this template, select individual values to be parameterized. At configuration push time, the parameters will be replaced with values from the HPE Server Automation Library, based on applicable inheritance rules.

```
<!DOCTYPE html>
<!--QCCR1K4344-->
<html xmlns:MadCap="http://www.madcapsoftware.com/Schemas/MadCap.xsd" lang="en-us" xml:lang="en-us" data-mc-search-type="Stem" data-mc-help-system-file-name="Default.xml" data-mc-path-
to-help-system="../../" data-mc-target-type="WebHelp2" data-mc-runtime-file-type="Topic" data-mc-preload-images="false" data-mc-in-preview-mode="false" data-mc-medium="non-print" data-
mc-toc-path="">
    <!-- saved from url=(0016)http://localhost -->
    <head>
        <meta charset="utf-8" />
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" /><title>SA Provisioning Help</title>
        <link href="../../Skins/Default/Stylesheets/TextEffects.css" rel="stylesheet" />
        <link href="../../Skins/Default/Stylesheets/Topic.css" rel="stylesheet" />
        <style>

*
```

Parameter Properties

Parameter name:

Type:

Display name:

Description:

Advanced

Conditional: □

Parameter is: ● Required ○ Optional

Loop variable: ○ Yes ● No

Ordered: □

Field Delimiter: WHITESPACE

Sequence type:

Sequence Delimiter:

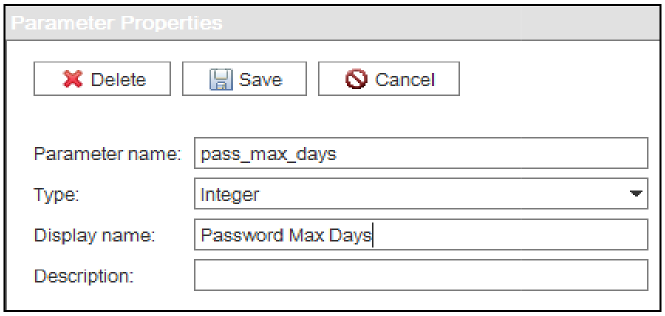
Range: From: To:

Parameter Properties panel

When you select a value in the Visual Editor and then click inside the **Parameter Name** field of the Parameter Properties panel, the fields in the Parameter Properties panel are populated based on the values in the underlying configuration file.

The parameter properties fields are editable in the user-friendly form view.

Parameter Properties form



Parameter name: pass_max_days

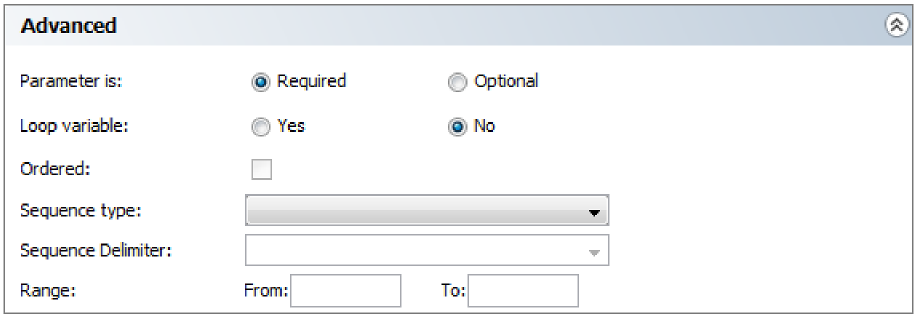
Type: Integer

Display name: Password Max Days

Description:

Advanced section of Parameter Properties: Use the Advanced section to specify additional options about the parameter, such as whether it is required (default) or optional, if it is a loop variable, an ordered list, a sequence (specify type and delimiter), or a range. The range is specified for a numeric type such as integer, decimal or port. (The numeric type can be part of a sequence.)

Advanced section of the Parameter Properties form



Advanced

Parameter is: ● Required ○ Optional

Loop variable: ○ Yes ● No

Ordered: □

Sequence type:

Sequence Delimiter:

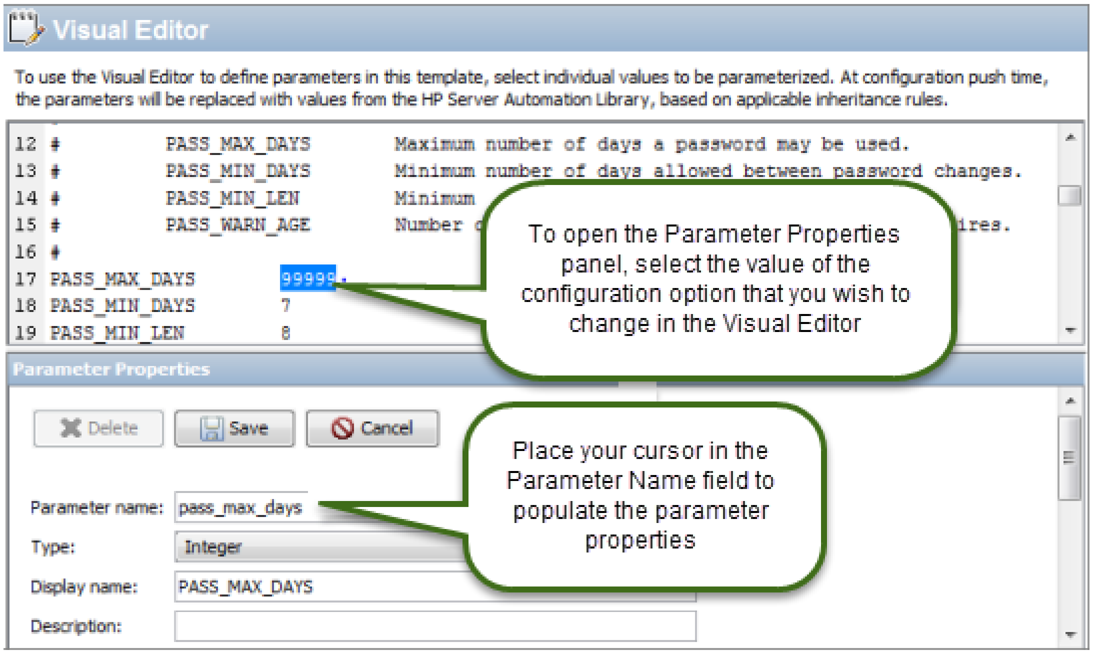
Range: From: To:

How the Visual Editor Interface Works

In the Visual Editor, select the value of configuration option that you wish to replace when you push the configuration and the form view appears.

The marked options populate the Parameter name, Display name, and Type (data type) fields in the Parameter Properties panel based on the data in the underlying configuration file. The form view allows you to modify the parameter details as desired.

Editing the configuration template in Visual Editor



To use the Visual Editor to define parameters in this template, select individual values to be parameterized. At configuration push time, the parameters will be replaced with values from the HP Server Automation Library, based on applicable inheritance rules.

```
#           PASS_MAX_DAYS       Maximum number of days a password may be used.
#           PASS_MIN_DAYS       Minimum number of days allowed between password changes.
#           PASS_MIN_LEN        Minimum
#           PASS_WARN_AGE       Number o                                      ires.
#
PASS_MAX_DAYS          99999.
PASS_MIN_DAYS          7
PASS_MIN_LEN           8
```

Parameter name: pass_max_days

Type: Integer

Display name: PASS_MAX_DAYS

Description:

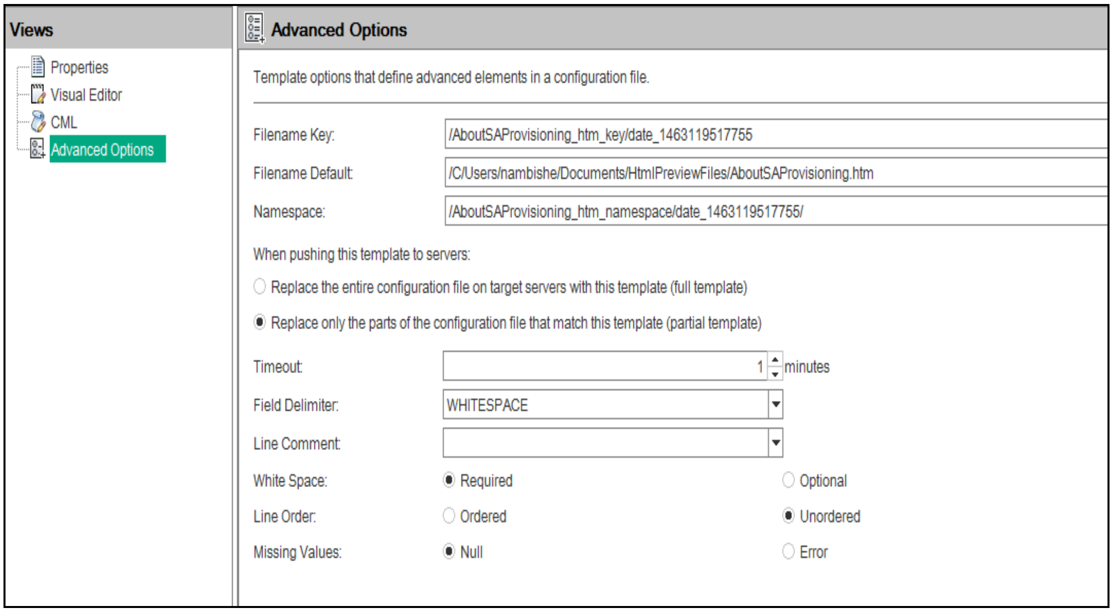
To open the Parameter Properties panel, select the value of the configuration option that you wish to change in the Visual Editor

Place your cursor in the Parameter Name field to populate the parameter properties

For instructions on editing the configuration template with the Visual Editor, see "Editing a configuration template with Visual Editor" on the next page.

Advanced options in Visual Editor

The Advanced Options view allows the user to modify global CML options in the template. To access this view, select **Advanced Options** in the Views pane of the template.



Advanced Options

Template options that define advanced elements in a configuration file.

Filename Key: /AboutSAProvisioning_htm_key/date_1463119517755

Filename Default: /C/Users/nambishe/Documents/HtmlPreviewFiles/AboutSAProvisioning.htm

Namespace: /AboutSAProvisioning_htm_namespace/date_1463119517755/

When pushing this template to servers:

○ Replace the entire configuration file on target servers with this template (full template)

● Replace only the parts of the configuration file that match this template (partial template)

Timeout: 1 minutes

Field Delimiter: WHITESPACE

Line Comment:

White Space: ● Required ○ Optional

Line Order: ○ Ordered ● Unordered

Missing Values: ● Null ○ Error

These values are populated by default and do not require user input, but they are editable:

- Defining the three mandatory fields: filename-key; filename-default and namespace
- Specifying whether the generated template will be a partial or full template (partial, by default, meaning only the specified values will be replaced.)
- The push timeout value and other CML formatting and parsing options.

> **Note:** Visual Editor Limitations: The Visual Editor editor only supports a portion of the CML content. If the CML is modified directly in a way that is not supported by the Visual Editor, a warning message will inform you of the incompatibility. You will have a choice of keeping your changes and disabling the Visual Editor, or abandoning your changes and staying with the Visual Editor.

Editing a configuration template with Visual Editor

In addition to basic instructions for the flow of the Visual Editor, this section has tips for editing a configuration template with the Visual Editor. For instructions on opening the visual editor, see "Creating a configuration template with Visual Editor" on page 261.

To edit a configuration template:

1. Open the Visual Editor.
   See"Importing and creating a template" on page 262 or "Creating a template from a working

configuration file" on page 262.

2. Define the parameters you want to push.
   See "Creating a simple parameter" below or "Creating a sequence parameter" on page 271 for examples.

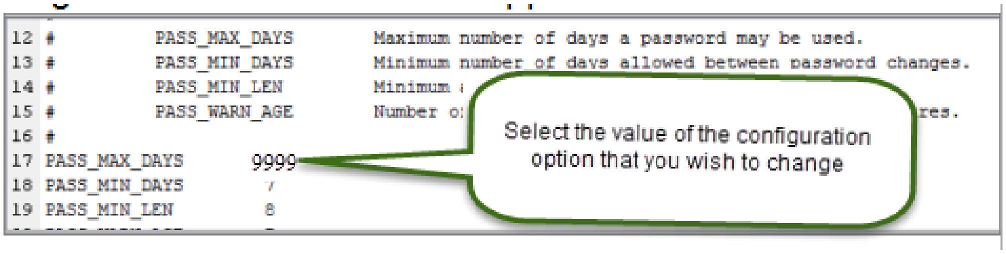
3. Review the **Properties** tab to specify the name of the template and the location where you want store it.

4. Click **Save** to save the template.

Creating a simple parameter

To create a simple parameter:

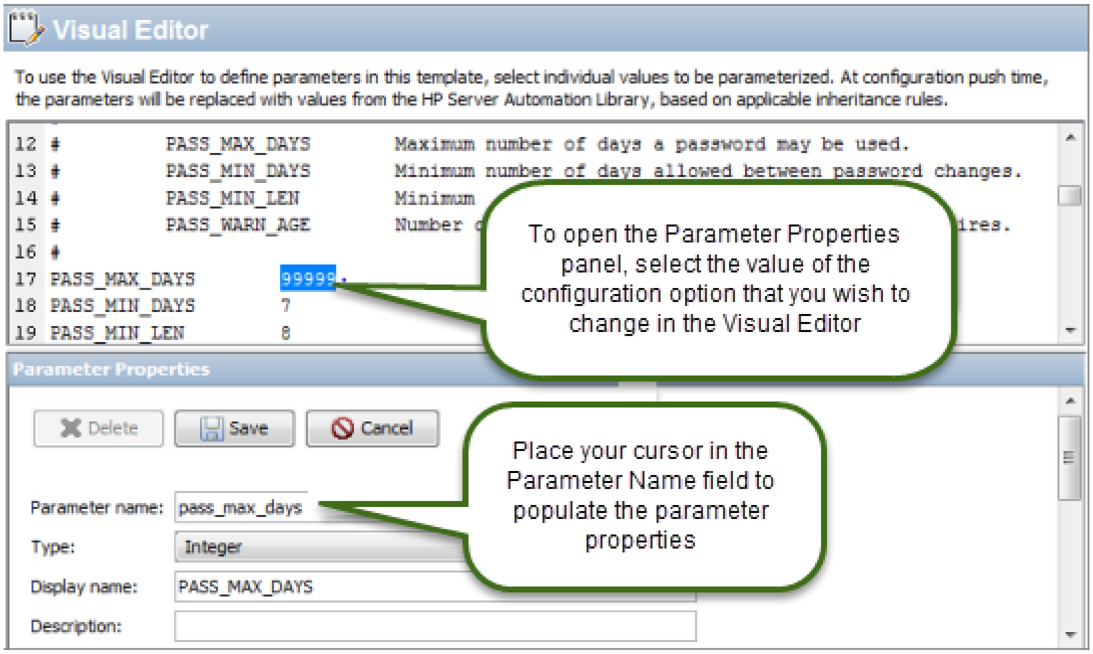
1. Select the configuration option that you wish to replace when you push the configuration and the form view appears.



```
#          PASS_MAX_DAYS       Maximum number of days a password may be used.
#          PASS_MIN_DAYS       Minimum number of days allowed between password changes.
#          PASS_MIN_LEN        Minimum :
#          PASS_WARN_AGE       Number o:                                          res.
#
PASS_MAX_DAYS       9999
PASS_MIN_DAYS          7
PASS_MIN_LEN           8
```

Select the value of the configuration option that you wish to change

2. In the Parameter Properties panel, click in the **Parameter name** field, and the marked options populate the **Parameter name**, **Display name**, and **Data Type** fields based on the underlying template.



To use the Visual Editor to define parameters in this template, select individual values to be parameterized. At configuration push time, the parameters will be replaced with values from the HP Server Automation Library, based on applicable inheritance rules.

```
#          PASS_MAX_DAYS       Maximum number of days a password may be used.
#          PASS_MIN_DAYS       Minimum number of days allowed between password changes.
#          PASS_MIN_LEN        Minimum
#          PASS_WARN_AGE       Number o                                         ires.
#
PASS_MAX_DAYS         99999
PASS_MIN_DAYS         7
PASS_MIN_LEN          8
```

To open the Parameter Properties panel, select the value of the configuration option that you wish to change in the Visual Editor

Parameter name: pass_max_days

Type: Integer

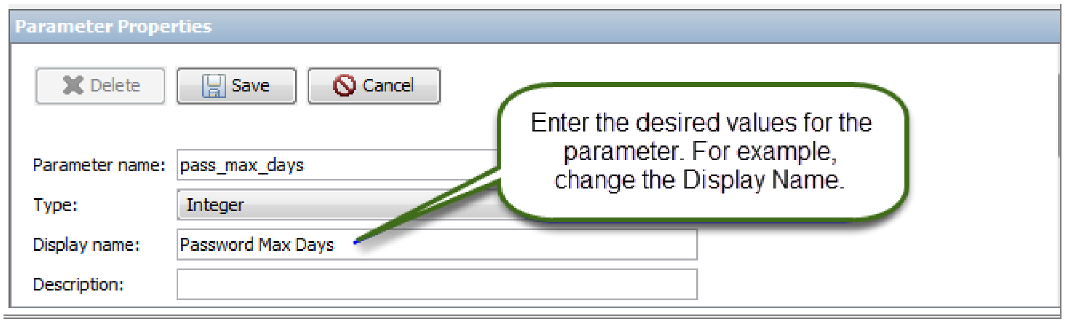
Display name: PASS_MAX_DAYS

Description:

Place your cursor in the Parameter Name field to populate the parameter properties

`

3. Enter the desired values for the **Parameter name** and **Display name** and click **Save** to save the parameter.



Parameter name: pass_max_days

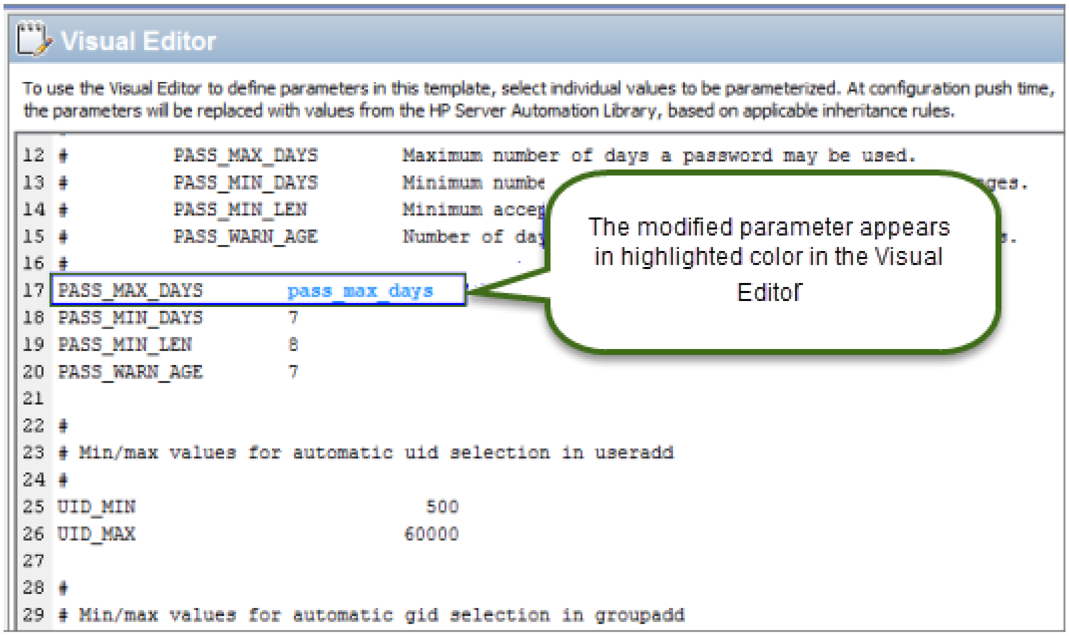
Type: Integer

Display name: Password Max Days

Description:

Enter the desired values for the parameter. For example, change the Display Name.

4. The newly created parameter will be highlighted in blue in the Visual Editor.



To use the Visual Editor to define parameters in this template, select individual values to be parameterized. At configuration push time, the parameters will be replaced with values from the HP Server Automation Library, based on applicable inheritance rules.

```
#          PASS_MAX_DAYS       Maximum number of days a password may be used.
#          PASS_MIN_DAYS       Minimum numbe                                   ges.
#          PASS_MIN_LEN        Minimum accep
#          PASS_WARN_AGE       Number of da                                   s.
#
PASS_MAX_DAYS          pass_max_days
PASS_MIN_DAYS          7
PASS_MIN_LEN           8
PASS_WARN_AGE          7

#
# Min/max values for automatic uid selection in useradd
#
UID_MIN                          500
UID_MAX                        60000

#
# Min/max values for automatic gid selection in groupadd
```

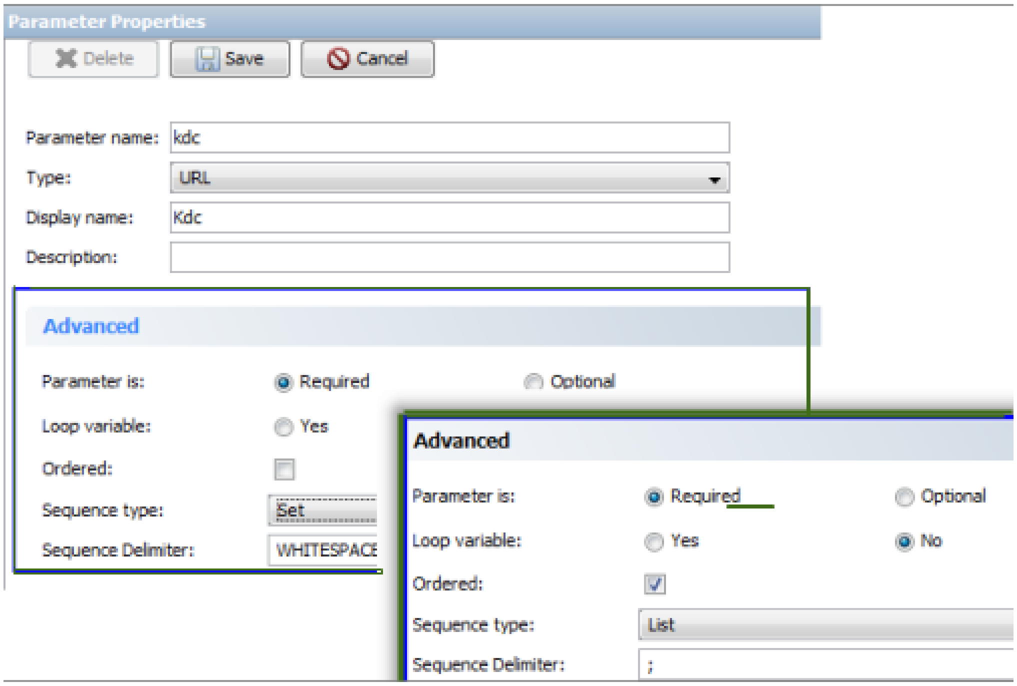
The modified parameter appears in highlighted color in the Visual Editor

Creating a sequence parameter

To create a sequence parameter:

1. Perform the same steps as when you create a simple parameter (see "Creating a simple parameter" on page 269).
2. In the Advanced section, specify a value in the **Sequence Type** field: Set or List.

   A List must hold the elements in a particular sequence, and a Set cannot have any duplicate elements.



Parameter name: kdc

Type: URL

Display name: Kdc

Description:

Advanced

Parameter is: ● Required ○ Optional

Loop variable: ○ Yes

Ordered: □

Sequence type: Set

Sequence Delimiter: WHITESPACE

Advanced

Parameter is: ● Required ○ Optional

Loop variable: ○ Yes ● No

Ordered: ☑

Sequence type: List

Sequence Delimiter: ;

3. Specify the Sequence Delimiter. This is the type of character that is used to separate the values in the list.

In the Visual Editor panel, the sequence parameter will be created in such a way that all the elements in the configuration file that match the parameter will be highlighted using the same parameter name.

Managing configuration files

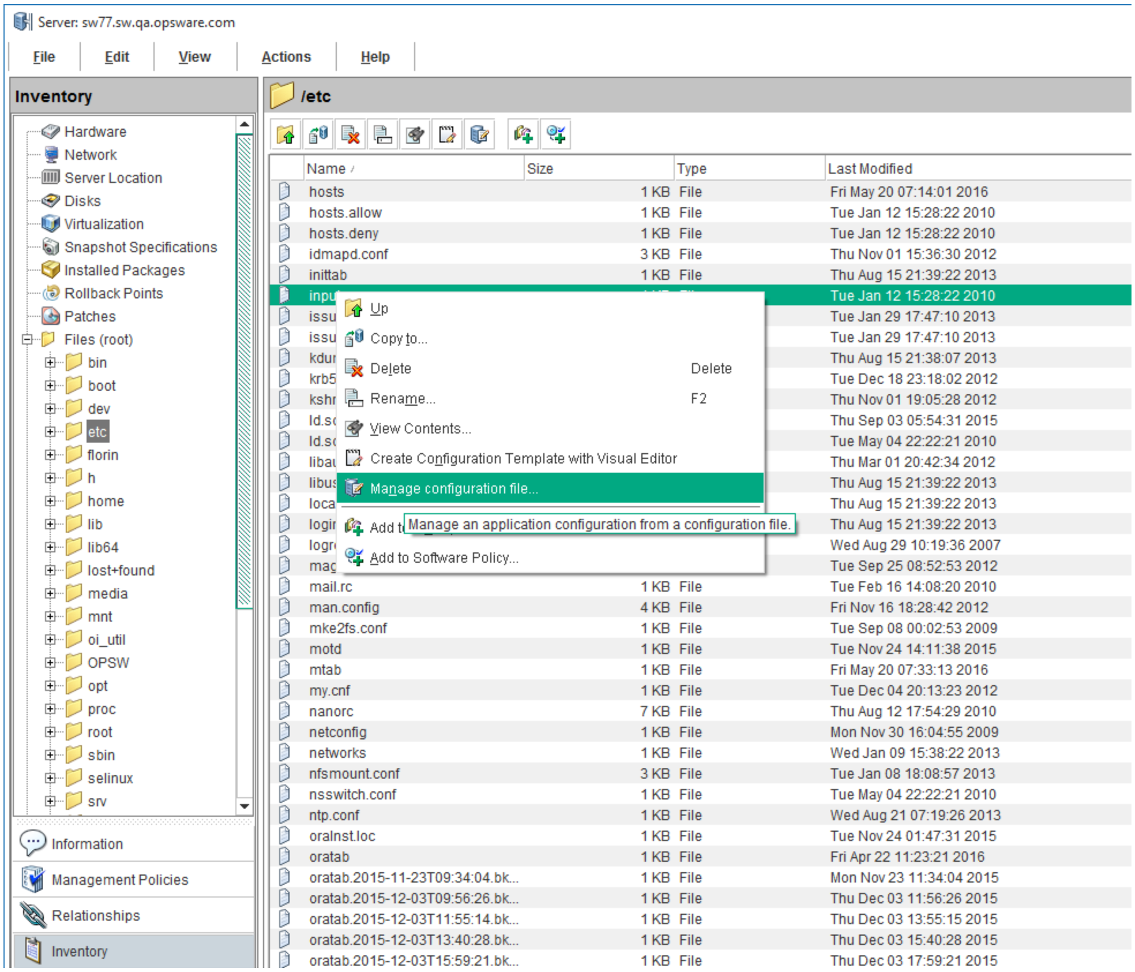
The Manage Configuration File feature enables you to quickly model a working configuration using an existing template.

> **Note:** You must have OGFS permissions to access the file system root directory in order to initiate this operation. You must also have right permission to your Home folder in order to complete this operation.

To model a working configuration using an existing templates:

1. Open a server browser:
   a. From the SA Client navigation pane, access the list of managed servers or device groups:
      i. Select **Devices** > **Servers** > **All Managed Servers** to view the server list.
      ii. Select **Devices** > **Device Groups** to view the device group list.
   b. From the content pane, select the server or device group you want to open.
   c. From the **Actions** menu, select **Open**.
2. Navigate to **Server Browser** > **Inventory** > **Files**. (This step requires that you have OGFS permissions.)

   When prompted choose the appropriate root path for her server file system (typically, root for UNIX or Administrator for Windows.)
3. Right-click on the configuration file and select **Manage Configuration File**.

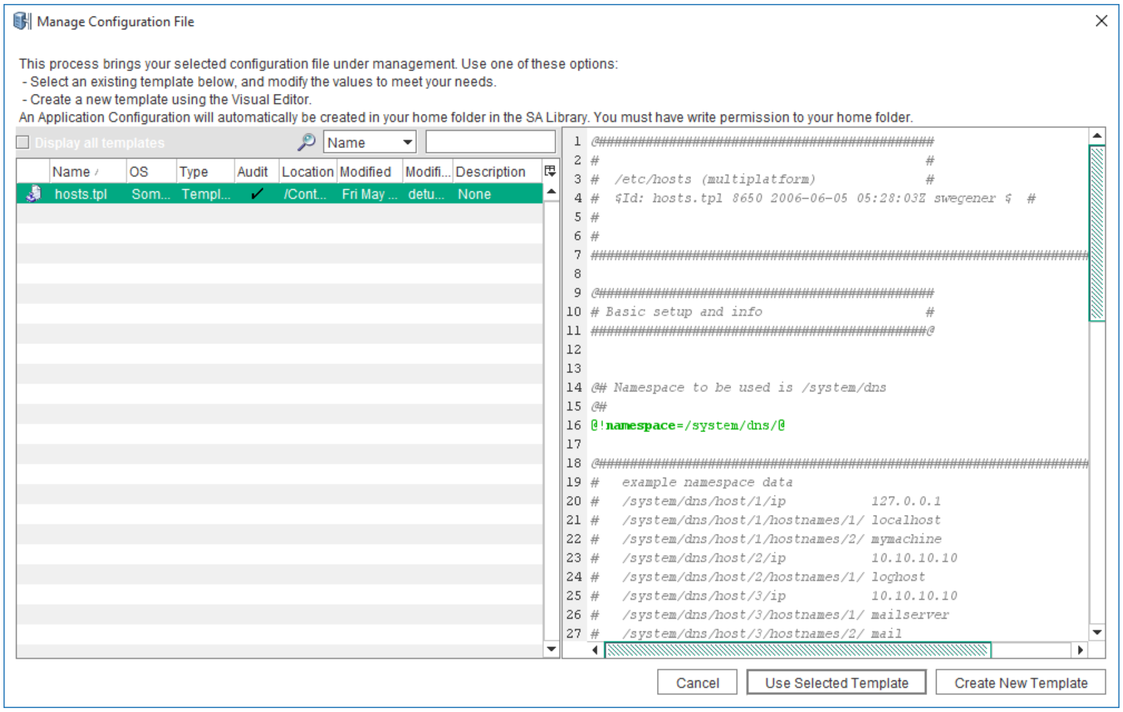


Name	Size	Type	Last Modified
hosts	1 KB	File	Fri May 20 07:14:01 2016
hosts.allow	1 KB	File	Tue Jan 12 15:28:22 2010
hosts.deny	1 KB	File	Tue Jan 12 15:28:22 2010
idmapd.conf	3 KB	File	Thu Nov 01 15:36:30 2012
inittab	1 KB	File	Thu Aug 15 21:39:22 2013
inpu			Tue Jan 12 15:28:22 2010
issu			Tue Jan 29 17:47:10 2013
issu			Tue Jan 29 17:47:10 2013
kdur			Thu Aug 15 21:38:07 2013
krb5			Tue Dec 18 23:18:02 2012
kshr			Thu Nov 01 19:05:28 2012
ld.sc			Thu Sep 03 05:54:31 2015
ld.sc			Tue May 04 22:22:21 2010
libau			Thu Mar 01 20:42:34 2012
libus			Thu Aug 15 21:39:22 2013
loca			Thu Aug 15 21:39:22 2013
logir			Thu Aug 15 21:39:22 2013
logr			Wed Aug 29 10:19:36 2007
mag			Tue Sep 25 08:52:53 2012
mail.rc	1 KB	File	Tue Feb 16 14:08:20 2010
man.config	4 KB	File	Fri Nov 16 18:28:42 2012
mke2fs.conf	1 KB	File	Tue Sep 08 00:02:53 2009
motd	1 KB	File	Tue Nov 24 14:11:38 2015
mtab	1 KB	File	Fri May 20 07:33:13 2016
my.cnf	1 KB	File	Tue Dec 04 20:13:23 2012
nanorc	7 KB	File	Thu Aug 12 17:54:29 2010
netconfig	1 KB	File	Mon Nov 30 16:04:55 2009
networks	1 KB	File	Wed Jan 09 15:38:22 2013
nfsmount.conf	3 KB	File	Tue Jan 08 18:08:57 2013
nsswitch.conf	1 KB	File	Tue May 04 22:22:21 2010
ntp.conf	1 KB	File	Wed Aug 21 07:19:26 2013
oraInst.loc	1 KB	File	Tue Nov 24 01:47:31 2015
oratab	1 KB	File	Fri Apr 22 11:23:21 2016
oratab.2015-11-23T09:34:04.bk...	1 KB	File	Mon Nov 23 11:34:04 2015
oratab.2015-12-03T09:56:26.bk...	1 KB	File	Thu Dec 03 11:56:26 2015
oratab.2015-12-03T11:55:14.bk...	1 KB	File	Thu Dec 03 13:55:15 2015
oratab.2015-12-03T13:40:28.bk...	1 KB	File	Thu Dec 03 15:40:28 2015
oratab.2015-12-03T15:59:21.bk...	1 KB	File	Thu Dec 03 17:59:21 2015

Manage an application configuration from a configuration file.

Existing templates in SA that match the configuration filename and path are listed as suggestions. Only templates that match the server platform are offered.

4. Select the template to view the contents in the preview pane.



This process brings your selected configuration file under management. Use one of these options:
- Select an existing template below, and modify the values to meet your needs.
- Create a new template using the Visual Editor.

An Application Configuration will automatically be created in your home folder in the SA Library. You must have write permission to your home folder.

□ Display all templates

Name	OS	Type	Audit	Location	Modified	Modifi...	Description
hosts.tpl	Som...	Templ...	✓	/Cont...	Fri May ...	detu...	None

```
@###########################################
#                                          #
#  /etc/hosts (multiplatform)              #
#  $Id: hosts.tpl 8650 2006-06-05 05:28:03Z swegener $  #
#
#
################################################################

@###########################################
# Basic setup and info                     #
###########################################@


@# Namespace to be used is /system/dns
@#
@!namespace=/system/dns/@

@###############################################################
#   example namespace data
#   /system/dns/host/1/ip           127.0.0.1
#   /system/dns/host/1/hostnames/1/ localhost
#   /system/dns/host/1/hostnames/2/ mymachine
#   /system/dns/host/2/ip           10.10.10.10
#   /system/dns/host/2/hostnames/1/ loghost
#   /system/dns/host/3/ip           10.10.10.10
#   /system/dns/host/3/hostnames/1/ mailserver
#   /system/dns/host/3/hostnames/2/ mail
```

Alternative options:

a. Choose “Display all templates” check box to expand the list to include all templates that match the server's platform.

b. Click **Cancel** to close this window and return to the servers file system directory.

c. Click **Create New Template** to create a new template based on the selected configuration file using the Visual Editor. (See "Creating a template from a working configuration file" on page 262 or "Editing a configuration template with Visual Editor" on page 268.)

   i. If you select Create New Template, when you save and close the template, you are prompted to indicate if you want an application configuration to be generated and an instance be attached the server.

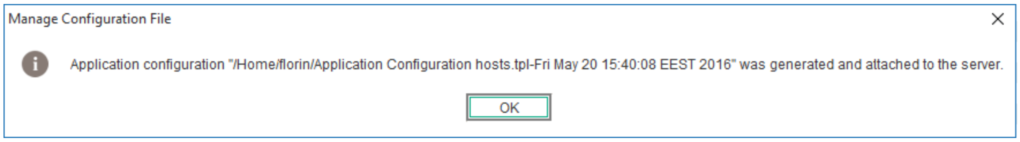
   ii. Optional: If you just want to save the template to revisit later, click **No**. The template is saved in the directory specified in the Properties tab.

   iii. Click **Yes** to instantly generate the application configuration instance and attach it to the server. The configuration instance appears in the Value Set Editor.

5. Click **Use Selected Template** to create a new application configuration using the selected template. (This step requires that you have permission to your Home folder.)

   The newly created configuration file will be placed in your Home folder in the SA Library and an instance of it will be attached to the managed server.

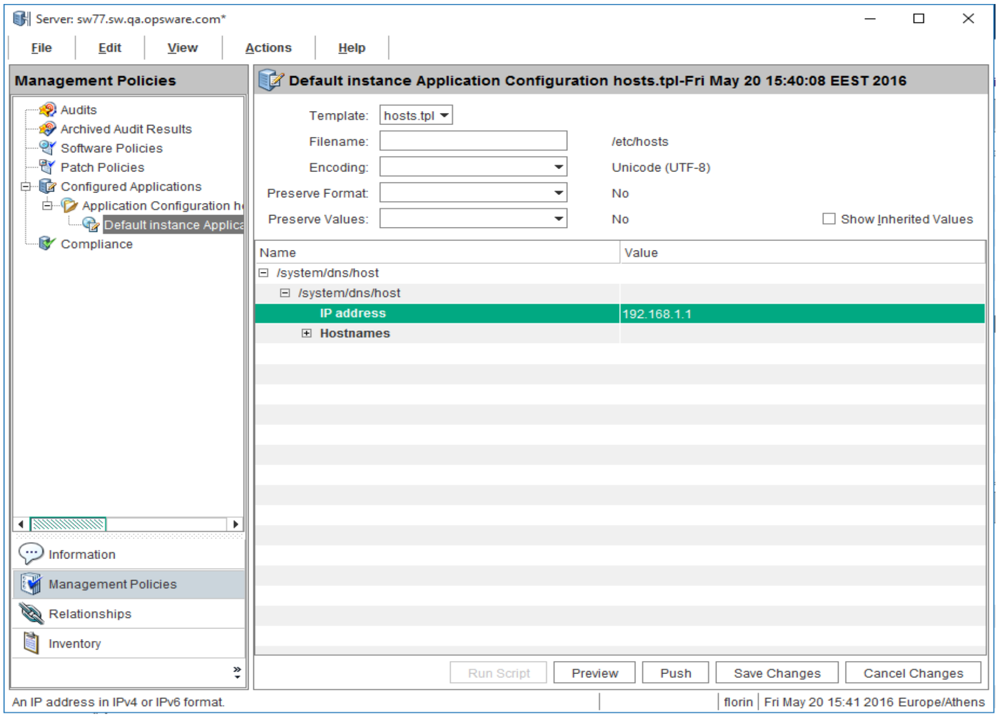
   A confirmation message appears advising that the configuration file was created and attached to the server.



Manage Configuration File

Application configuration "/Home/florin/Application Configuration hosts.tpl-Fri May 20 15:40:08 EEST 2016" was generated and attached to the server.

6. Click **OK**. The configuration instance appears in the Value Set Editor.



Server: sw77.sw.qa.opsware.com*

Default instance Application Configuration hosts.tpl-Fri May 20 15:40:08 EEST 2016

Template: hosts.tpl

Filename: /etc/hosts

Encoding: Unicode (UTF-8)

Preserve Format: No

Preserve Values: No

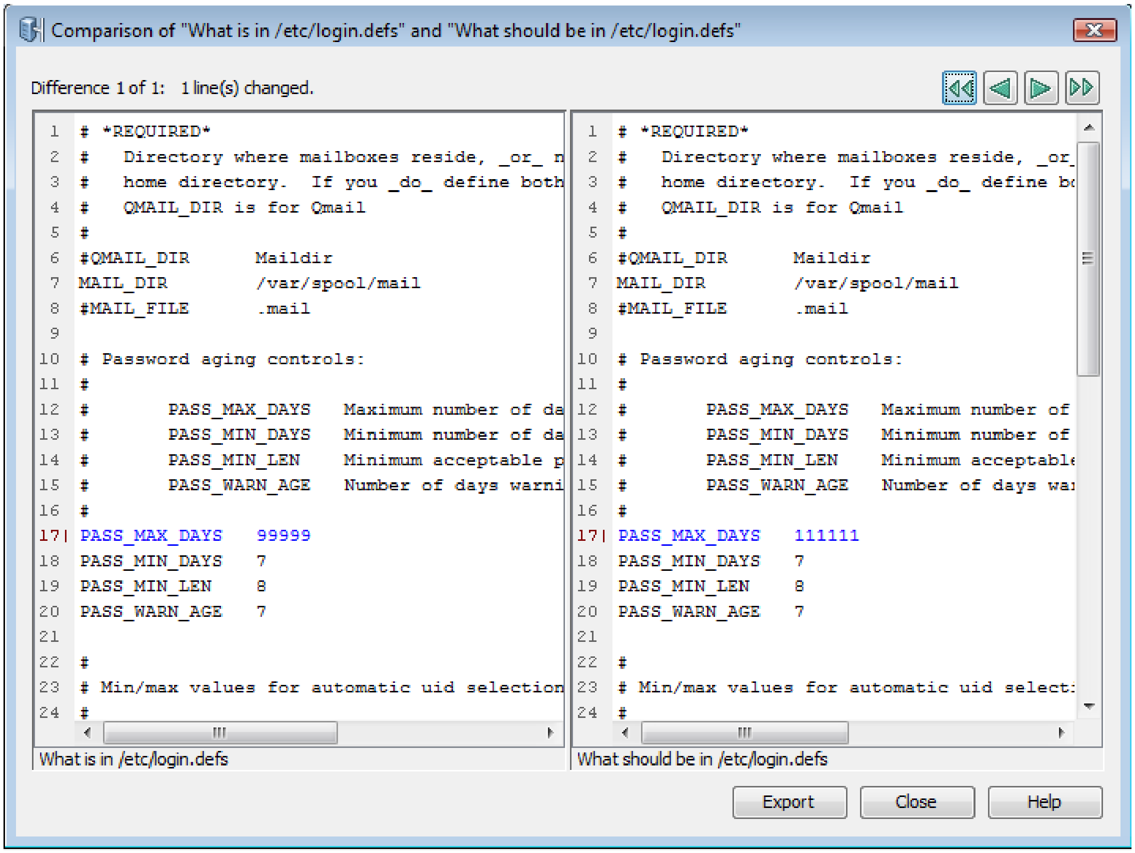
□ Show Inherited Values

Name	Value
⊟ /system/dns/host	
⊟ /system/dns/host	
IP address	192.168.1.1
⊞ Hostnames	

An IP address in IPv4 or IPv6 format. | florin | Fri May 20 15:41 2016 Europe/Athens

7. Enter the value set data for this instance (see "Setting values in the Value Set Editor" on page 232).

8. (Optional) Click **Preview** to preview the change compared to the current value side by side. Click **Close** to return to the Application Configuration instance view.



Comparison of "What is in /etc/login.defs" and "What should be in /etc/login.defs"

Difference 1 of 1: 1 line(s) changed.

```
# *REQUIRED*
#   Directory where mailboxes reside, _or_ n
#   home directory.  If you _do_ define both
#   QMAIL_DIR is for Qmail
#
#QMAIL_DIR      Maildir
MAIL_DIR        /var/spool/mail
#MAIL_FILE      .mail

# Password aging controls:
#
#       PASS_MAX_DAYS   Maximum number of da
#       PASS_MIN_DAYS   Minimum number of da
#       PASS_MIN_LEN    Minimum acceptable p
#       PASS_WARN_AGE   Number of days warni
#
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   7
PASS_MIN_LEN    8
PASS_WARN_AGE   7

#
# Min/max values for automatic uid selection
#
```

What is in /etc/login.defs

```
# *REQUIRED*
#   Directory where mailboxes reside, _or_
#   home directory.  If you _do_ define bo
#   QMAIL_DIR is for Qmail
#
#QMAIL_DIR      Maildir
MAIL_DIR        /var/spool/mail
#MAIL_FILE      .mail

# Password aging controls:
#
#       PASS_MAX_DAYS   Maximum number of
#       PASS_MIN_DAYS   Minimum number of
#       PASS_MIN_LEN    Minimum acceptable
#       PASS_WARN_AGE   Number of days wa
#
PASS_MAX_DAYS   111111
PASS_MIN_DAYS   7
PASS_MIN_LEN    8
PASS_WARN_AGE   7

#
# Min/max values for automatic uid select:
#
```

What should be in /etc/login.defs

9. Click **Push** to push the change to the server.

Editing CML or XML in a template

You can edit the CML or XML of your template in the Content view. The template editor provides editing operations and syntax highlighting as described below.

To edit the CML or XML of your template:

1. From the SA Client navigation pane, select **Library** and then select the **By Type** tab.
2. Locate and open the Application Configuration node. Open the Templates node. Open the operating system group and select the operating system where the template file is located. Note that a template can apply to multiple operating systems.
3. Select a template.

4. Select the **Actions** > **Open** menu or right-click and select the Open menu or press the Enter key. This displays the Templates screen showing the selected template.
5. Select the Content view. This displays the contents of your template.
6. Enter your CML or XML text directly in the template editor.

   The CML syntax is highlighted in colors to make reading it easier:

   - Green: CML instructions are displayed in green. CML key words are in bold green.
   - Blue: Variables that will be replaced by actual values from the value set are displayed in blue.
   - Black: Fixed text is displayed in black.
   - Gray: Comments are displayed in gray.

   The CML editor also provides the following editing operations:

   - Right-click in the CML text and use the cut, copy and paste operations.
   - The Actions menu provides find, replace, undo and redo operations.
   - The Validate button and the **Actions** > **Validate** menu check the syntax of your template and report any errors.

For more information on CML and XML, see "CML Reference" on page 373 and "Managing XML configuration files" on page 298.

Importing and validating a template file

You can write a CML template or an XML template with a text editor and import it into the SA Library for use in an Application Configuration. You can also validate the template from SA before importing it (see "Configuration templates and script templates" on page 227).

For configuration files on Windows servers which are encoded in UTF-8, the first three characters of the configuration file might contain a Byte Order Mark (BOM). If you import this file into an Application Configuration Template, the BOM will appear in the template after the file is imported. If you do not want this BOM to be included in the Application Configuration Template, remove it after you upload the configuration file into the template.

UTF-16 encoding is not supported in the SA Client.

To validate and import a template file:

1. From the SA Client select the Navigation pane, select Library then select the By Type tab.
2. Locate and open the Application Configuration node. Open the Templates node. Open the operating system group and select the operating system where the template file will be used. Note that a template can apply to multiple operating systems. You can specify this in a later step.
3. Select the **Actions** > **Validate Template...** menu.
4. Locate and select your template file, select the appropriate encoding and select Open. SA checks the syntax of your template and reports the results. If there are errors in the file, correct them and revalidate.
5. Select the **Actions** > **Import Template...** menu.
6. Locate and select your template file, select the appropriate encoding and select Open. Note that UTF-16 encoding is not supported in the SA Client. SA imports the template and displays the Template screen.
7. Follow the steps under "Creating a configuration template" on page 259 starting from step 4.

Viewing configuration template sources

You can view the contents of your configuration template and view its CML or XML source. This can be helpful to understand which list merging modes have been set in the template before you push the Application Configuration to a server. For information on Application Configuration sequence merge modes, see "Sequence aggregation" on page 417.

To view the source for a configuration template:

1. From the SA Client navigation pane, select Library then select the By Type tab.
2. Locate and open the Application Configuration node. Open the Templates node. Open the operating system group and navigate to the operating system where the template file is. Note that a template can apply to multiple operating systems.
3. Select a configuration template and select **Actions** > **Open**.
4. Select the Content view to display the CML or XML contents of the configuration template.

Adding or removing templates from an Application Configuration

An Application Configuration contains one or more templates.

To add or remove a template from an Application Configuration:

1. From the SA Client navigation pane, select Library and then select the **By Type** tab.
2. Locate and open the Application Configuration node. Open the Configurations node. Open the operating system group and navigate to the operating system where the Application Configuration is. Note that an Application Configuration can apply to multiple operating systems.
3. Select an Application Configuration and select **Actions** > **Open**.
4. Select the Configuration Values view.
5. To add a template to the application configuration, select **Actions** > **Add** or select the "+" button. Select the desired template and select **OK**.

   The templates must be applicable to all the platforms that the configuration is (that is, the OSs for the configuration must be a subset of those for the templates).

   The customer setting of the folder containing the template must include the customer setting of the application configuration object. Otherwise the template will not be included in the list of available templates. For more information on folder settings, see "Folder Permissions" in the SA Administering.
6. To remove a template, select the template and select **Actions** > **Remove** or select the "-" button.
7. Select **File** > **Save** to save your changes.

Specifying the template order in the Application Configuration

An Application Configuration can contain one or more configuration templates and related scripts. You can specify the order of the templates in your application configurations. The templates will be pushed to managed servers in the order in which they appear in the application configuration. For example, you may need to apply changes to certain configuration files before others.

The order of execution of scripts in an application configuration is determined by the script type: data-manipulation, pre-install, post-install or post-error. The order of the scripts in the application configuration is irrelevant. For more information, see "Types of Application Configuration scripts" on page 244.

To specify the order of the templates in your Application Configuration:

1. From the SA Client navigation pane, select Library and then select the By Type tab.
2. Locate the Application Configuration node and open it. Open the Configurations node. Open the operating system group and navigate to the operating system where the Application Configuration is. Note that an Application Configuration can apply to multiple operating systems.
3. Select an Application Configuration and select **Actions** > **Open**.
4. Select the **Configuration Values** view. This displays all the configuration templates and scripts (if any) in the Application Configuration. Notice that the templates and scripts are numbered in order.
5. To reorder a template or script, select it then select **Actions** > **Move Up** or **Actions** > **Move Down** or select the up-arrow and down-arrow icons.
6. Select **File** > **Save** to save your changes.

Creating a template from a script

To include a script in an application configuration object you must copy the script into a CML template then import the template into the application configuration object. For details on CML, see "CML Reference" on page 373. See "Running scripts with Application Configurations" on page 244.

The following example shows a simple Unix shell script that performs a touch command and an echo command:

```
#/bin/sh
touch abc.txt
echo abc &gt;&gt;abc.txt
The following example shows this Unix shell script converted to CML:
@###########################################################
# /tmp/simple-script/TouchABC.sh                          #
# Version 0.1                                             #
# Author: &lt;name&gt;                                              #
###########################################################@
@!namespace=/simple-script-namespace/@
```

```
@!filename-key="/TouchABC"@
@!filename-default=/tmp/simple-script/TouchABC.sh@
#/bin/sh
touch abc.txt
echo abc &gt;&gt;abc.txt
```

To create a template from this script:

1. Create a template as described in "Creating a configuration template" on page 259. Use the CML version of your script as the contents of the template.
2. Set the Type field to the appropriate script type. For the above example, you would set the Type to “Unix .SH script”. Other supported script types are listed in "Types of Application Configuration scripts" on page 244.
3. Set the Parser Syntax field to “CML Syntax” since all scripts must be written in CML syntax.
4. Set the remaining fields as described in "Creating a configuration template" on page 259.
5. Select **File > Save**.
6. Select **File > Close.**
7. Add your template to an application configuration object as described in " Adding or removing templates from an Application Configuration" on page 279.
8. After adding your template to an application configuration object, open the application configuration object.
9. Select the **Configuration Values** view.
10. Select the script template and right-click to display the menu.
11. Select the script type, which specifies when the script will run: Data-manipulation, Pre-install, Post-install or Post-error.
12. Select **File > Save**.
13. Select **File > Close.**

Managing non-text configurations by running a data manipulation script

With SA, you can manage non-text configurations by creating a data-manipulation script that extracts the non-text configuration data and places it in a text file. The text file can then be manipulated by SA the same as any other text configuration file and placed back into its original form. For a description of script types, see "Types of Application Configuration scripts" on page 244.

Some examples of non-text configuration data are:

- An SQL database with a data manipulation script could run some SQL queries and place the data in a text file.
- An IIS server’s configuration with a data-manipulation script that reads the metabase information into a text file.
- A binary file with a data manipulation script that extracts the values from the binary file and places them into a text file.

Running data manipulation scripts manually

The Run Script button allows you to execute a data manipulation script associated with an application configuration and to prepare a target configuration file on a managed server so its values can be imported into a value set.

To run a data manipulation script manually:

1. From the SA Client navigation pane, select the Devices tab.
2. Select either All Managed Servers or Device Groups. Navigate to the desired server or device group.
3. Select a server or device group and select the **Actions** > **Open** menu.
4. If you selected a server, perform the following steps. If you selected a device group, skip to the next step.
    a. Select the Management Policies tab.
    b. Open the Configured Applications node in the navigation pane. This displays all the application configurations attached to the server.
    c. Open the application configuration node you want to push. This displays the value sets for the application configuration.
    d. Select the application configuration instance you want to push.
5. If you selected a device group, perform the following steps.
    a. Open the Configured Applications node in the navigation pane. This displays all the application configurations attached to the servers and a list of all the servers.
    b. Open the Servers node. This displays all the servers in the device group.

c. Open the particular server node where you want to run the data manipulation script. This displays the application configuration attached to that server.

d. Open the application configuration node you want to push. This displays the value sets for the application configuration.

e. Select the value set you want to push. You must select a value set at the server instance level.

6. With a value set at the server instance level selected, select the Run Script button. This displays the confirmation dialog.
7. Select Yes to run the data-manipulation script on the server.
8. Once the data-manipulation script has run, extracted the configuration data and placed it into a text file, you can manage the text file the same as any other configuration file and place the data from the text file back into its original form.

Attaching an Application Configuration to a server or device group

After you have created an Application Configuration, added all the necessary configuration templates and scripts and edited its default values, you must attach it to a server or public device group. Once you attach an Application Configuration to a server or group of servers, you must push the application configuration to the servers as described in "Pushing Application Configurations" on page 287.

> - The customer setting of the folder containing the application configuration must include the customer setting of the managed servers where you intend to push the application configuration. For more information on folder settings, see “Folder Permissions” in the SA Administering. For more information about servers and customers, “Customer Accounts” in the Using.
> - You can only attach Application Configurations to individual servers or to public device groups. You cannot attach them to private device groups.

Attaching an Application Configuration to a single server

Shortcut: **Devices** tab > select **Servers** > **All Managed Servers** > right-click> **Attach** > **Application Configuration** > choose a configuration file > click **OK**.

To attach an Application Configuration **to a single server:**

1. From the SA Client navigation pane, select the **Devices** tab.
2. Select **Servers** > **All Managed Servers**.
3. From the Content pane, select a server.
4. Select the **Actions** > **Open** menu.
5. Select the **Management Policies** tab, then select **Configured Applications**.
6. Select the Installed Configurations tab.
7. Select the **Actions** > **Add Configuration... menu**.
8. In the Select Application Configuration screen, select the Application Configuration that you want to attach to the managed server.
   You can use the search tool to search by a specific criteria such as name, date last modified, and so on.

   > **Note:**
   > The customer setting of the folder containing the application configuration must include the customer setting of the managed servers where you intend to push the application configuration. For more information on folder settings, see “Folder Permissions” in the SA 10.50 Administration Guide. For more information about servers and customers, “Customer Accounts” in the SA 10.50 User Guide.

9. Select **OK** to attach the Application Configuration to the server.
10. Select the Save Changes button.
11. You can now set the Application Configuration’s values for that server. For more information on setting up the Application Configuration values, see " Value sets" on page 229.

Attaching an Application Configuration to a device group

To attach an Application Configuration **to a device group:**

You can only attach Application Configurations to public device groups. You cannot attach Application Configurations to private device groups.

1. From the SA Client navigation pane, select the Devices tab.
2. Open the Device Groups node and navigate to a public device group.
3. From the Content pane, select the desired device group.

4. Select the **Actions** > **Open** menu.
5. In the device group screen, select the Configured Applications view.
6. Select the **Actions** > **Add Configuration** menu.
7. In the Select Application Configuration dialog box, select an Application Configuration.
   Use the search tool to search by a specific criteria such as name, date last modified and so on.
8. Enter an Instance Name. This is the name of the value set at the group instance level. For more information, see "Setting values at the group instance level" on page 241.
9. Select **OK** to attachthe Application Configuration to the device group. The specified configuration file can be pushed to all servers in the group.
10. Set the values to be pushed to the servers. For more information on setting values, see" Value sets" on page 229.

Detaching an Application Configuration from a server or device group

To detach an application configuration from a server you must open the server and remove all the instances of the application configuration at the server instance level. To detach an application configuration from a device group you must open the device group and remove all the instances at the group instance level. The following sections provide details.

Detaching an Application Configuration from a server

To detach an application configuration from a server, you must remove all the instances at the server instance level as described below. For more information, see "Setting values at the server level" on page 243.

To detach an application configuration from a server:

1. Open the server in the SA Client. The application configuration must be attached to the server. See " Attaching an Application Configuration to a server or device group" on page 283.
2. Select the Management Policies tab on the left.
3. Open the Configured Applications node in the Management Policies pane on the left.

4. Open the desired application configuration node under the Configured Applications node. This displays all the instances of the application configuration at the server instance level.
5. Under the application configuration node, select one of the instances.
6. Select the **Actions** > **Remove Configuration** menu or right-click and select the **Remove Configuration** menu. This removes the selected instance.
7. Repeat "Under the application configuration node, select one of the instances." above and "Select the Actions > Remove Configuration menu or right-click and select the Remove Configuration menu. This removes the selected instance." above for each instance. When you remove the last instance, the application configuration is detached from the server.
8. Click **Save Changes**.

Detaching an Application Configuration from a device group

To detach an application configuration from a device group, you must remove all the instances of the application configuration at the group instance level as described below. For more information, see "Setting values at the group level" on page 240.

To detach an application configuration from a device group:

1. From the SA Client navigation pane, select the Devices tab.
2. Open the Device Groups node and navigate to a public device group.
3. From the Content pane, select the desired device group.
4. Select the **Actions** > **Open** menu.
5. In the device group screen, select the Configured Applications view and open the Configured Application node. This displays the application configurations attached to the device group.
6. Open the desired application configuration node under the Configured Applications node. This displays the value sets at the group instance level.
7. Under the desired application configuration node, select one of the application configuration instances at the group instance level.
8. Select the **Actions** > **Remove Configuration** menu or right-click and select the **Remove Configuration** menu. This removes the selected instance.

9. Repeat step 7 and step 8 above for each instance at the server instance level. When you remove the last instance, the application configuration is detached from the server.
10. Click **Save Changes**.

Pushing Application Configurations

Anytime you change values in a value set, to merge those changes with the configuration file on the target server you must push the application configuration to the server. For more information, see "Pushing Application Configurations to servers" on page 246.

To push application configuration changes to a server or group of servers:

1. From the SA Client navigation pane, select the **Devices** tab.
2. Select either All Managed Servers or Device Groups. Navigate to the desired server or device group.
3. Select a server or device group and select the **Actions** > **Open** menu.
   - If you selected a server, select the Management Policies tab.
   - If you selected a device group, skip to the next step.
4. Open the Configured Applications node in the navigation pane and select the application configuration instance you want to push.

   You can optionally preview the changes that will be made on an individual server by selecting the **Preview button**. The comparison screen shows any differences. Select **Close** when you are finished.
5. When you are ready to apply the changes to the server or servers, select **Push**.
6. In the Push Configurations screen, verify the Application Configuration and the value set to be pushed.
   To accept the remaining defaults for Scheduling, Notifications and Job Status, click **Start Job**. Otherwise click **Next** to continue reviewing the wizard options.
7. In the Scheduling pane, specify when you want the Application Configuration to be pushed. You can use the Scheduling pane to schedule the job to run in the future or to run at regular recurring intervals such as weekly or monthly.
   To accept the remaining defaults for Notifications and Job Status, click **Start Job**. Otherwise click **Next** to continue reviewing the wizard options.

8. In the Notifications pane, optionally specify one or more email addresses and a ticket identifier. For each recipient, select the options for when to send an email notification:
   - On Success: sends email to recipient if the job succeeds.
   - On Failure: sends email to recipient if the job fails.
   - On Termination: sends email to recipient if the job is terminated.
     - Termination occurs when you stop an actively running job via the End Job action.
     - This notification does *not* apply to jobs that are cancelled before they are run.

   To accept the remaining defaults for Job Status, click **Start Job**. Otherwise click **Next** to continue reviewing the wizard options.

9. Click **Start Job**.

   After the job has started, you can view its status by selecting the Jobs and Sessions tab then Job Logs on the main SA Client screen.

   You can also perform any of the following optional actions:
   - Click **Export** to export the job status results to a text file.
   - Click **End Job** to stop the job. See "Stopping a push configuration job" below.
   - Click **Close** to close the window. To view job status later, click **Job Status** from the SA Client navigation pane, and then double-click on the job to view details.

Stopping a push configuration job

You can terminate a push configuration job that is actively running. For example, you may need to stop a job that is producing erroneous results or will run beyond an allotted maintenance window.

To maintain job integrity, some steps in the push configuration job flow are non-cancellable. When you stop the job, the Job Status window will indicate which steps completed and which were skipped.

To stop an active application configuration push job:

1. From the Job Status window, click **End Job**. (This button only appears if the job is in progress.)
2. The End Job warning dialog will be displayed advising you how job termination works:
   - The job will not initiate work on any additional servers
   - If work has started on a server, the job will only skip steps that can be safely cancelled

- The Job Status will indicate the steps that were completed or skipped
- If the job terminates successfully, the final job status will be “Terminated”

3. Click **OK** to confirm that you wish to terminate the job. The Job Status window displays the progress of the termination.

   The job status will be Terminated. The server status will be Cancelled. The task statuses will be Succeeded or Skipped.

4. When the termination is complete, you can also view the job in the SA Client Job Log.

   From the SA Client navigation pane, click **Jobs and Sessions**. The Job Logs view appears with your job listed with Terminated status.

Modifying push timeout values

By default, when you push an Application Configuration, the default timeout value is ten minutes, plus one minute for each template inside the Application Configuration. Each template in that Application Configuration appends its timeout to the base timeout for the Application Configuration.

For example, if you have an Application Configuration that contains three templates, the default timeout value for the entire Application Configuration is 13 minutes. If you pushed the template and the entire push took longer than 13 minutes, the push will time out and the operation is cancelled, including any changes that were already made.

To extend a template’s timeout value, you can use the CML timeout tag for individual templates inside the Application Configuration. The CML timeout tag syntax is as follows:

```
@!timeout=1@
```

Valid values are 0-999, in minutes.

If the Application Configuration times out in the middle of a push, all changes to the target file of the push are backed out and the operation is cancelled.

For details on the CML timeout tag, see the "CML Reference" on page 373.

Scheduling an Application Configuration push

You can schedule an Application Configuration push to run immediately, in the future or on a recurring schedule, such as daily, weekly, or monthly. To schedule an Application Configuration push, follow the steps listed under "Pushing Application Configurations" on page 287 but when you get to the Scheduling step, enter the frequency and time when you want the push to occur.

Once you have scheduled the job, you can view its status by selecting the Jobs and Sessions tab the Recurring Schedules on the main SA Client screen.

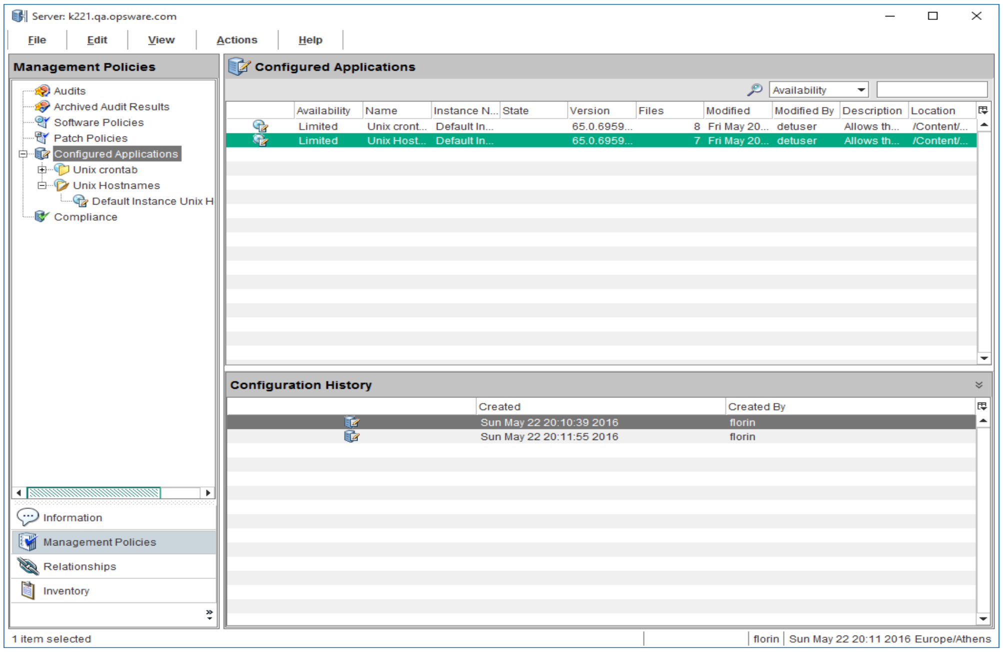
Restoring a configuration file to a previous state

Every time you push an Application Configuration to a server, the configuration files are saved in a configuration push history list. At any time, you can restore an Application Configuration to a previous state in the history list.

To restore an Application Configuration **to a previous state:**

1. From the SA Client navigation pane, select the Devices tab.
2. Select All Managed Servers and locate the desired server.
3. Select a server and select the **Actions** > **Open** menu.
4. Select the Management Policies tab.
5. Select the Configured Applications node in the navigation pane. This displays all the application configurations installed on the server.

6. Select a configuration to view the Configuration History in the detail pane. This displays all the push jobs that have run on the server.

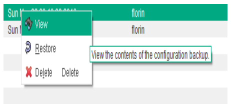


	Availability	Name	Instance N...	State	Version	Files	Modified	Modified By	Description	Location
	Limited	Unix cront...	Default In...		65.0.6959...	8	Fri May 20...	detuser	Allows th...	/Content/...
	Limited	Unix Host...	Default In...		65.0.6959...	7	Fri May 20...	detuser	Allows th...	/Content/...

	Created	Created By
	Sun May 22 20:10:39 2016	florin
	Sun May 22 20:11:55 2016	florin

1 item selected | florin | Sun May 22 20:11 2016 Europe/Athens

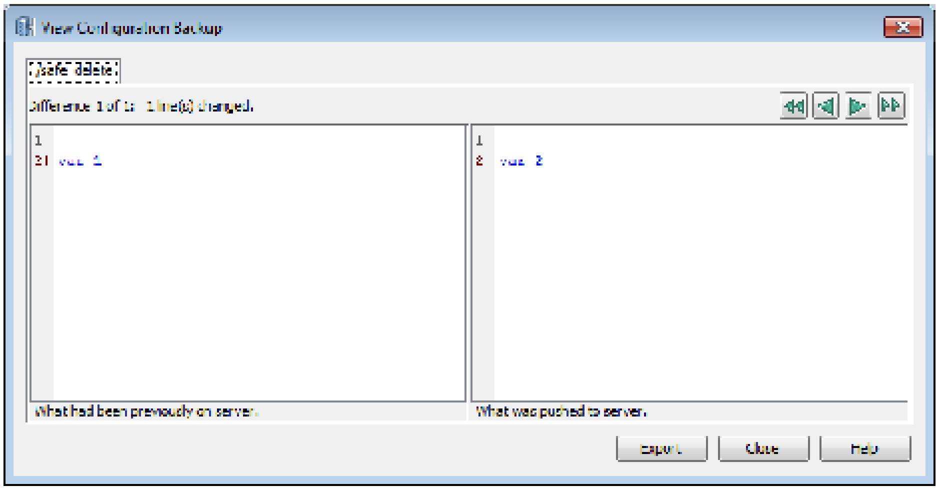
7. Right-click one of the historical instances, and select **View** to view a snapshot.



View the contents of the configuration backup.

The View Configuration Backup window appears in a split pane for comparison:

- ○ The left pane displays the content of the pushed file before the snapshot, which is essentially a backup of the existing file if there was one.
- ○ The right pane displays the content of the file that was pushed in the selected snapshot.



Difference 1 of 1: 1 line(s) changed.

```
1
2 var 1
```

```
1
2 var 2
```

What had been previously on server.

What was pushed to server.

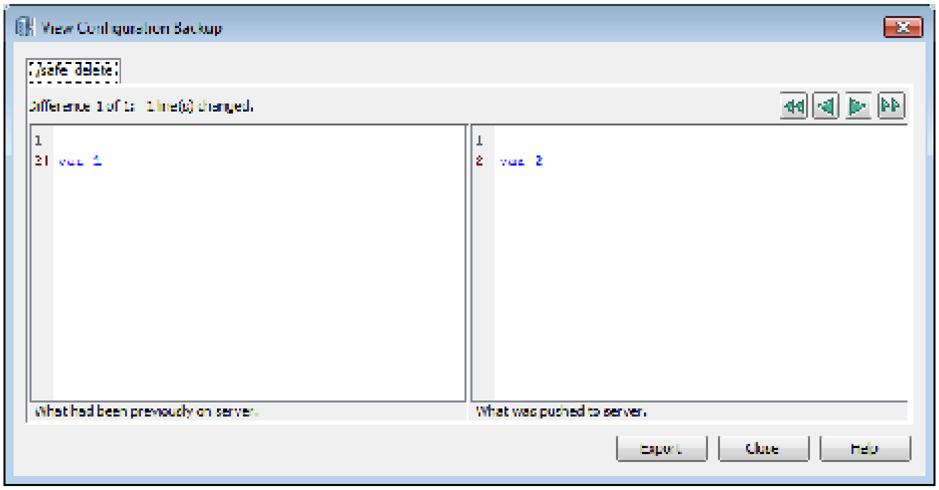
8. Select the line in the history that you want to restore and click **Restore**.
   This displays the Restore Configurations wizard.

9. Verify the server and history instance to restore and click **Next**.

10. Select the restore type:
    - *Previous to selected push (undo)* restores the configuration file to the values immediately before the selected history instance. That is, it undoes the selected instance.
    - *Following selected push (redo)* restores the configuration file to the values of the selected history instance. That is, it repushes the selected instance.

11. Click **Next**.

12. In the Preview step, click **Preview** to compare the change to the current instance on the server.
    The View Configuration Backup window appears in a split pane for comparison:
    - The left side displays what is currently on the server at a given location.
    - The right side contains what will be restored to that location, based on the “Restore Type” that was selected in the job.



Difference 1 of 1: 1 line(s) changed.

```
1
2| val 1
```

What had been previously on server.

```
1
2  val 2
```

What was pushed to server.

This preview gives you an option to review the results before committing the restoration:
- To cancel, close the window and click **Cancel** from the job window.
- To proceed, close the window and proceed to the next step.

13. Click **Start Job**. This restores the configurations on the server to the selected history instance.

14. After the job has started, you can view its status by selecting the Jobs and Sessions tab then Job Logs on the main SA Client screen.

Searching and filter job results

You can search and filter the results of a push or restore job. This is useful when your job runs on a large number of servers. Note that you can search and filter any kinds of SA jobs, not just push and restore jobs.

To search and filter results:

1. From the SA Client navigation pane, select the **Jobs and Sessions** tab.
2. Select **Job Log**. This displays a list of jobs that have run on your servers.
3. Select a job from the job list.
4. Select the **Actions** > **Open** menu. This displays the details of the selected job.
5. In the job screen, select Job Status. This displays the results of the job.
6. Select any step in the Actions column.
7. On your keyboard, type Ctrl-F. This displays the Find tool for the steps in the job. Enter the text you want to search for in the text box and use the buttons to search and highlight.
8. Select the detailed text in the lower box.
9. On your keyboard, type Ctrl-F. This displays the Find tool for the details of the job results. Enter the text you want to search for in the text box and use the buttons to search and highlight.
10. To remove the Find tool, select the text box and type Esc on your keyboard.

Comparing a configuration template with a target configuration file - Preview

Before you push an application configuration to a server, you can compare the proposed application configuration with the target configuration file on the server with the Preview button. You can select a server from your managed servers or you can select a server in a device group.

Selecting a server from your managed servers

To compare the values in a configuration template with an actual configuration file on a server:

1. From the SA Client navigation pane, select the **Devices** tab.
2. Select **Servers** > **All Managed Servers**.
3. From the Content pane, select a server.
4. Select the **Actions** > **Open** menu.
5. Select the **Management Policies** tab.
6. Open the **Configured Applications** node to display all the application configurations attached to the server.
7. Open the desired application configuration node to display all the instances of that application configuration.
8. Select an instance of the application configuration.
9. In the content pane, select a template from the drop-down list.
10. Click **Preview**. This compares the generated configuration file generated from the template and value set with the actual configuration file on the server and displays both files side by side with color coding to make it easier to interpret.
    - **Green**: This indicates that new information has been added.
    - **Blue**: This indicates that information has been modified.
    - **Red**: This indicates that information has been deleted.
    - **Black**: This indicates no changes.

Selecting a server from a device group

To compare the values in a configuration template with an actual configuration file on a server in a device group:

1. From the SA Client navigation pane, select the **Devices** tab.
2. Select **Device Groups** > **Public to display your public device groups.**
3. Navigate to the desired device group and from the Content pane, select a public device group.

4. Select the **Actions** > **Open** menu.
5. Open the **Configured Applications** node to display all the application configurations attached to the device group and the servers in the device group.
6. Open the Servers node to display all the servers in the device group.
7. Open the desired server mode to display the application configurations attached to that server.
8. Open the desired application configuration node to display all the instances of the application configuration.
9. Select an instance of the application configuration.
10. In the content pane, select a template from the drop-down list.
11. Click **Preview**. This compares the generated configuration file generated from the template and value set with the actual configuration file on the server and displays both files side by side with color coding to make it easier to interpret.
    - **Green**: This indicates that new information has been added.
    - **Blue**: This indicates that information has been modified.
    - **Red**: This indicates that information has been deleted.
    - **Black**: This indicates no changes.

Comparing configuration templates

To compare two configuration template**s:**

1. From the SA Client navigation pane, select Library then select the By Type tab.
2. Locate and open the Application Configuration node. Open the Templates node. Open the operating system group and navigate to the operating system where the template files are. Note that a template can apply to multiple operating systems.
3. Select a configuration template.
4. Hold down the Ctrl key on the keyboard and select the second template.
5. Right click and select the Compare menu. The Comparison screen displays the differences between the two files. Use the arrows in the upper right of the screen to navigate through the two files. The following colors indicate the differences:
    - **Blue** indicates information that is different between the two templates.
    - **Red** indicates information that has been removed.

- **Green** indicates information that has been added.
- **Black** indicates identical text.

6. When you are finished viewing the differences, select **Close**.

Localizing configuration file element names

You can display the names of configuration file elements in your local language in the value set editor of the SA Client. This can be useful when your system administrators who will be specifying value sets use a different language.

Localization files represent locale-specific resource files, where you define localized strings for configuration template elements. These localized strings display in the value set editor.

Localization templates are used only in the value set editor in the SA Client and are ignored during a push.

Creating a localization file

To localize the names of configuration file elements in the SA Client, you must first create a localization template in the SA Library.

To create a localization file:

1. From the SA Client navigation pane, select Library and then select the By Type tab.
2. Locate and open the Application Configuration node. Open the Templates node. Open the operating system group and select the operating system where the template file will be used. Note that a template can apply to multiple operating systems. You can specify this in a later step.
3. Select the **Actions** > **New** menu. This displays the Templates screen where you define the properties of the template.
4. Select the Properties view and enter a description of the localization template file as well as the following information:
    - **Name**: Specify the name of the localization file. The naming convention for localization template files requires that localization files end with `.<locale>`. The values for `<locale>` are defined by ISO-639 and are two-letter lower case codes for representing the names of languages. For more information on ISO 639, see http://www.loc.gov/standards/iso639-2/php/code_list.php

- For example, .es represents Spanish, .en represents English, .fr represents French, .zh represents Chinese, and .hi represents Hindi.
- **Location**: Specify where in the **SA** Library you want to store the localization template.
- **Version**: The version can be any string you use for tracking changes to the template. The version is not incremented automatically.
- **Type**: Specify Localization file as the type.
- **Availability**: Use this setting to keep track of which localization files are tested and ready to be used and which are not yet tested or are deprecated. This setting does not change what can be done with the localization file. You can use this field value as a search criteria.

5. Select Content view.
6. Enter your localization instructions directly in the template editor.
   - The format for display text localization instructions is:

     `printable/<name space>/<variable> <localized string>`

     Where `printable` is the key word indicating that the line is a localization instruction for display text.

     `<name space>` is the name space where the desired variable is stored in the database.

     `<variable>` is the variable defined in a configuration template.

     `<localized string>` is the text that will be displayed in the value set editor instead of the full name space and variable name.

   - The format for tool tip localization instructions is:

     `description/<name space>/<variable> <localized string>`

     Where `description` is the key word indicating that the line is a localization instruction for tool tip text.

     `<name space>` and *`<variable>`* are the same as above.

     `<localized string>` is the text that will be displayed in the value set editor when someone hovers the mouse over the item.

   See "Editing CML or XML in a template" on page 276 for the editing operations and syntax highlighting.
7. Select **Validate** to parse the syntax and check for errors.
8. Select **File** > **Save** to save your template.

9. Add your localization template to an application configuration object as described in "Applying a localization template" below.

Applying a localization template

After creating a localization template, you apply it to an application configuration so the configuration file elements display in your local language.

To apply a localization template to an application configuration:

1. Open your application configuration object as follows.
   a. From the SA Client navigation pane, select Library and then select the By Type tab.
   b. Locate and open the Application Configuration node. Open the Configurations node. Open the operating system group and navigate to the operating system where the Application Configuration is. Note that an Application Configuration can apply to multiple operating systems.
   c. Select an Application Configuration and select **Actions** > **Open**.
2. Select the Properties view.
3. Under the Localization Files in the content pane, select the “+” button**.**
4. In the Select Localization Template screen, select a localization template.
5. Select **OK**.
6. Select **File** > **Save**. This applies your localization template and displays the configuration file elements in the language specified by the localization template. Whenever anyone displays the value set editor for the application configuration, the localized strings will be displayed rather than the raw name space and variable names.

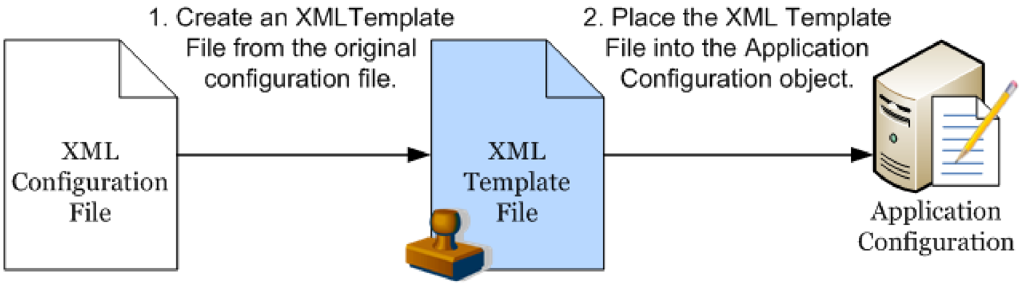
Managing XML configuration files

With SA you can manage XML configuration files from a central location and propagate changes across multiple servers in your data center. You can create, edit, and store configuration file values to ensure that the XML configuration files on your managed servers are correct. You can manage XML files that use a DTD as well as XML files that do not.

This section discusses how XML configuration templates are structured so you can manage generic (non-DTD) XML files, as well as XML files that reference a DTD. Since XML is well-structured, SA needs only a minimum amount of information to be able to model and manage XML-based configuration files.

To manage XML configuration files you first need to create a **template** file for your XML configuration file. After creating the template, you must add it to an **application configuration** object so you can manage, edit, and make changes to the native configuration files on managed servers.

Configuration files


1. Create an XMLTemplate File from the original configuration file.
2. Place the XML Template File into the Application Configuration object.
XML Configuration File
XML Template File
Application Configuration

The following section describes a simple XML file and shows how to create an application configuration for a non-DTD based XML file and one for a DTD-based XML file.

Also see the following examples:

- "XML Tutorial 1 - Creating a non-DTD XML configuration template" on page 309.
- "XML Tutorial 2 - Creating an XML-DTD configuration template" on page 315.

Example: Travel manager application and XML configuration file

This section describes an example web application that uses a simple XML file to control its configuration and shows how to create an application configuration to manage that file.

Travel Manager is a web application designed to help people manage their travel by performing such tasks as booking hotels, rental cars, tracking expenses, and so on. Travel Manager uses the MySQL Relational Database Management System (RDMS) as the repository for user data and some of the application’s configuration data.

Since the Travel Manager is designed to be deployed over many different networks, each with a different database server, it is important to provide flexibility in the information used to connect to the MySQL server. The application is designed to retrieve connection information from an XML configuration file, `mysql.xml`.

With application configurations, you can set the configuration file values necessary for accessing the local MySQL database. For example, the user name and password used to open a connection to the database may be different for each installation of the Travel Manager application. Modifications to these values can be made to the configuration file without requiring a recompilation of the Travel Manager application code.

Only four values in the file `mysql.xml` are required for the Travel Manager to be able to connect to the local MySQL database, each of which is represented as an element in the application’s XML file:

- **Host**: Host name of the server on which the MySQL RDMS has been installed.
- **Name**: Name of the database on the host server.
- **User**: User name credentials used to open a connection to the database.
- **Password**: Password necessary to open a connection to the database.

Contents of the Travel Manager mysql.xml file

The following is an example of the Travel Manager `mysql.xml` configuration file:

```
<?xml version="1.0" ?>
<db-config>
  <db-host>localhost</db-host>
  <db-name>wrightevents</db-name>
  <db-user>root</db-user>
  <db-password>hp-pass</db-password>
</db-config>
```

Contents of the Travel Manager mysql.xml DTD-based XML file

The following is an example of the Travel Manager `mysql.xml` configuration file that references a DTD:

```
<?xml version="1.0"?>
<!DOCTYPE db-config PUBLIC "-//Williams Events//Travel Manager//EN" "mysql2.dtd">
<db-config>
       <db-host>localhost</db-host>
       <db-name>wrightevents</db-name>
       <db-user>root</db-user>
```

```
        <db-password>hp-pass</db-password>
</db-config>
```

Non-DTD XML configuration templates

You can create a non-DTD based XML configuration template written as a single XML comment with three pieces of required information that enables the template to extract and store values from a target XML file:

- `ACM-NAMESPACE`: Defines the location where values read from the target XML file on the managed server will be stored in the database. The name space must be unique and the path must start with a forward slash (/).
- `ACM-FILENAME-DEFAULT`: Defines the default absolute path of the target XML configuration file on the managed server.
- `ACM-FILENAME-KEY`: Defines the location in the name space where the target XML configuration file name will be stored.

When you set a configuration template's properties to use XML syntax, the labels displayed in the value set editor are the same as the tag names for the each corresponding element inside the XML file.

For a full list of template settings for XML templates, see "XML configuration template settings" on page 307.

For information on setting the parser syntax to XML for a configuration template, see "Creating a configuration template" on page 259.

Non-DTD XML configuration template for mysql.xml

The following example shows the XML configuration template based on the `mysql.xml` file. The template file is named to `mysql.tpl` to indicate it is a template file.

```
<!--

ACM-NAMESPACE = /TravelManager/

ACM-FILENAME-KEY = /files/TravelManager

ACM-FILENAME-DEFAULT = /var/www/html/we/mysql.xml

ACM-TIMEOUT = 1

-->
```

This example shows that the XML configuration template references the target XML file (`/var/www/html/we/mysql.xml`), so it can be parsed by the application configuration parser, and its values read and stored in the SA Library.

The `mysql.tpl` configuration template contains the following required information:

- `ACM-NAMESPACE`: Defines the location where values read from the `mysql.xml` file on the managed server will be stored in the database. The name space must be unique and the path must start with a forward slash (/).
- `ACM-FILENAME-DEFAULT`: Defines the default absolute path of the `mysql.xml` file on the managed server.
- `ACM-FILENAME-KEY`: Defines the location in name space where the `mysql.xml` file name will stored.
- `ACM-TIMEOUT`: (Optional) Represents the number of minutes that are added to the configuration template’s default timeout value of ten minutes during a push.

The default timeout value for an entire application configuration is ten minutes plus the timeout for each configuration template inside the application configuration. So if this template were the only template inside an application configuration (which has a ten minute timeout), and this value is set to 1, the overall timeout value for the entire application configuration when pushed would be eleven minutes.

DTD-based XML configuration templates

An XML-DTD configuration template is actually just an XML DTD with some application configuration options defined in the comments. Since the DTD standard defines the syntax and layout of an XML file, there is no need to redefine that syntax in another language.

For DTD-based XML files, XML-DTD configuration templates require the same three basic attributes required for a generic XML file — `ACM-NAMESPACE, ACM-FILENAME-DEFAULT, and ACM-FILENAME-KEY –` plus three other attributes:

- `ACM-DOCTYPE`: Defines the name of the root element in the XML file. The root element follows the opening `<!DOCTYPE` declaration found in the target XML configuration file.
- `ACM-DOCTYPE-SYSTEM-ID`: Defines the name of the associated DTD file on the managed server. This value is typically found in the XML configuration file as the SYSTEM attribute in the DOCTYPE element.
- `ACM-DOCTYPE-PUBLIC-ID`: Defines a string that represents a public identifier of the XML document. This value is typically found in the XML configuration file as the PUBLICID attribute of a DOCTYPE element.

For a complete list of all XML configuration file attributes, see "XML configuration template settings" on page 307.

XML-DTD configuration template for mysql.xml

The following is an example of the configuration template created for the Travel Manager DTD-based XML file.

```
<!--
    ACM-FILENAME-KEY = /files/TravelManager
    ACM-FILENAME-DEFAULT = /var/www/html/we/mysql.xml
    ACM-NAMESPACE = /TravelManager/
    ACM-TIMEOUT = 1
    ACM-DOCTYPE = db-config
    ACM-DOCTYPE-SYSTEM-ID = mysql.dtd
    ACM-DOCTYPE-PUBLIC-ID = -//Williams Events//Travel Manager//EN
    -->
    <!ELEMENT db-config (db-host,db-name,db-user,db-password)>
    <!ELEMENT db-host (#PCDATA)>
    <!ELEMENT db-name (#PCDATA)>
    <!ELEMENT db-user (#PCDATA)>
    <!ELEMENT db-password (#PCDATA)>
```

In this example, the DOCTYPE attributes reference specific XML and DTD information that enables the parser to extract information from both the DTD file and the referenced XML file.

Specifically, the DTD-based XML configuration templates must contain the following information:

- `ACM-DOCTYPE`: The root node of the targeted XML file. For `mysql.xml`, the root node is `dbconfig`.
- `ACM-DOCTYPE-SYSTEM-ID`: The name of the DTD file being targeted by the configuration template. In the example of `mysql.xml`, the DTD being used is named `mysql.dtd`.
- `ACM-DOCTYPE-SYSTEM-ID`: The public ID of the XML file.

Customize XML DTD element display

There are two optional settings you can add to your XML-DTD configuration template that allow you to customize how elements from the target XML-DTD configuration file are displayed in the value set editor in the SA Client. The `ACM-PRINTABLE` and `ACM-DESCRIPTION` optional settings allow you to control the names of elements as they appear in the SA Client:

- ACM-PRINTABLE: Defines the label for each element from the XML file that is displayed in the value set editor when the XML-DTD template is shown in the SA Client.
- ACM-DESCRIPTION: Defines mouse-over text when a user moves a mouse pointer over the field defined in ACM-PRINTABLE in the value set editor in the SA Client.

Explicit versus positional display settings

You can set the printable and description values for attributes and elements inside the XML-DTD configuration template in either of two ways: positionally or explicitly.

- With *positional* definitions, ACM-PRINTABLE and ACM-DESCRIPTION are inserted directly after the element or attribute they are describing inside the XML-DTD configuration template.
- With *explicit* definitions, ACM-PRINTABLE and ACM-DESCRIPTION can be defined anywhere in the template.

```
<!ELEMENT db-config (db-host,db-name,db-user,db-password)>
<!--
ACM-PRINTABLE = database configuration
ACM-DESCRIPTION = The db-config element specifies the data structure that contains
the information needed to connect to a database.
-->

<!ELEMENT db-host (#PCDATA)>
<!--
ACM-PRINTABLE = database hostname
ACM-DESCRIPTION = The db-host element specifies the name of the host computer (the
server) on which the database engine is running.
-->

<!ELEMENT db-name (#PCDATA)>
<!--
ACM-PRINTABLE = database name
ACM-DESCRIPTION = The db-name element specifies the name of the database.
-->

<!ELEMENT db-user (#PCDATA)>
<!--
ACM-PRINTABLE = database user
ACM-DESCRIPTION = The db-user element specifies the user identification used to
connect to the database.
-->

<!ELEMENT db-password (#PCDATA)>
<!--
```

```
ACM-PRINTABLE = database password
ACM-DESCRIPTION = The db-password element specifies the password used to connect to
the database.
-->
```

Add positional custom display settings

The positional method for adding element tables and mouse-over text to an XML template is to add a comment immediately after the element or attribute definition you want to define, and in that comment set the ACM-PRINTABLE and ACM-DESCRIPTION values. In other words, for either XML elements or attributes, you can specify a label and a mouse-over description for the label directly.

In the following example, each XML element from mysql.xml defines a ACM-PRINTABLE and ACM-DESCRIPTION setting immediately after each element in the XML-DTD template.

Add explicit custom display settings

The explicit method for adding settings to an XML-DTD template allows you to define ACM-PRINTABLE and ACM-DESCRIPTION values anywhere in the configuration template by specifying the element name with the ACM-ELEMENT tag and optionally the attribute name with the ACM-ATTRIBUTE tag.

For this method the ACM-ELEMENT tag is required, even when defining printable and description values for attributes, because attributes are always associated with specific elements.

Once you have set the ACM-ELEMENT and the ACM-ATTRIBUTE tags, you can also set the ACM-DESCRIPTION and ACM-PRINTABLE tags within the same comment block. You should only use one definition per comment-block. In other words, define a ACM-PRINTABLE and ACM-DESCRIPTION for a single element, and then start a new comment block for the next element.

The ACM-ELEMENT tag and ACM-ATTRIBUTE tag (when applicable) should be defined before the ACM-PRINTABLE and ACM-DESCRIPTION tags.

For example, to customize the mysql.tpl template, you would construct the template as follows:

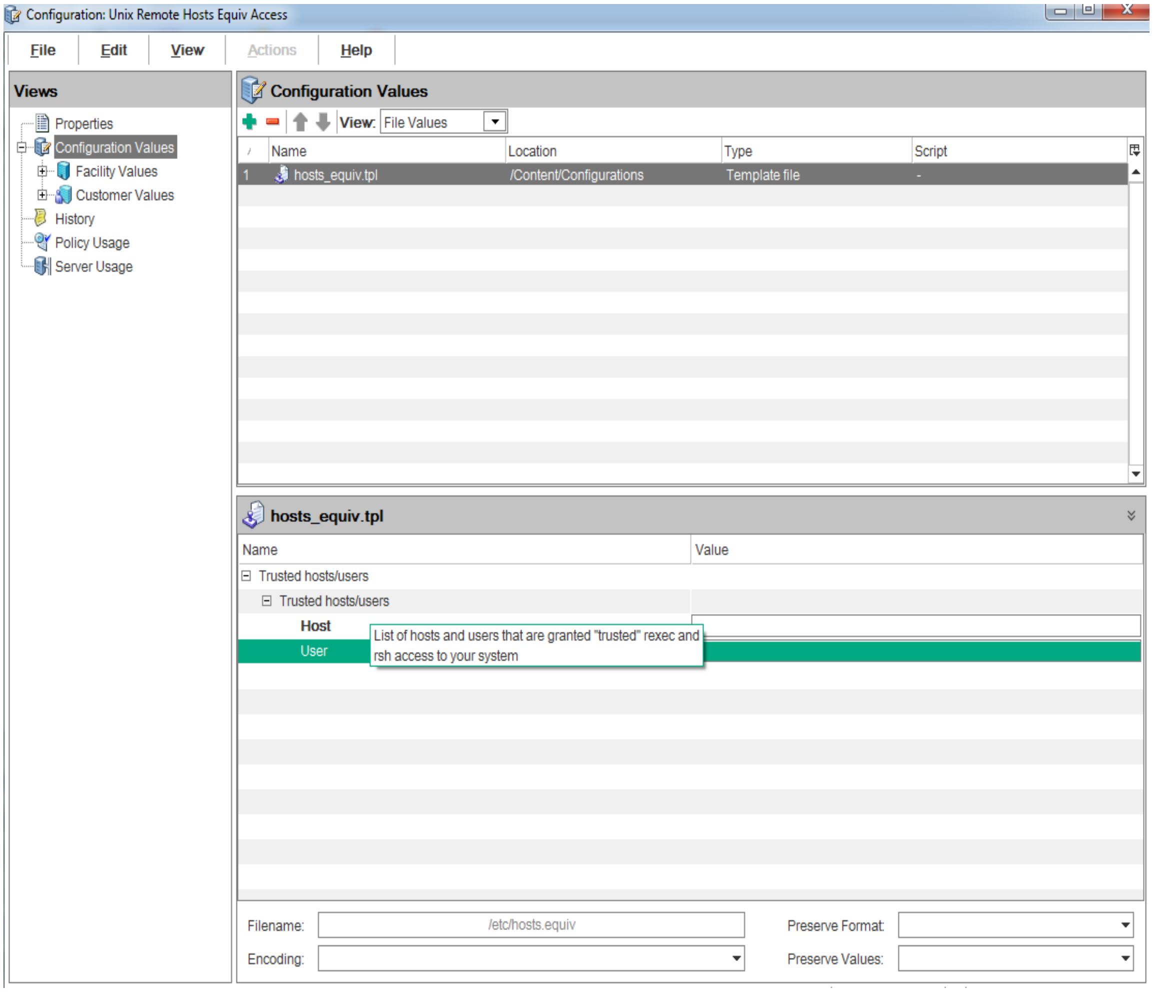
```
<!--
ACM-TIMEOUT = 1
ACM-FILENAME-KEY = /files/TravelManager
ACM-FILENAME-DEFAULT = /var/www/html/we/mysql2.xml
ACM-NAMESPACE = /TravelManager/
ACM-DOCTYPE = db-config
ACM-DOCTYPE-SYSTEM-ID = mysql.dtd
ACM-DOCTYPE-PUBLIC-ID = -//Williams Events//Travel Manager//EN
-->

<!ELEMENT db-config (db-host,db-name,db-user,db-password)>
```

```
<!ELEMENT db-host     (#PCDATA)>
<!ELEMENT db-name     (#PCDATA)>
<!ELEMENT db-user     (#PCDATA)>
<!ELEMENT db-password (#PCDATA)>

<!--
ACM-ELEMENT = db-config
ACM-PRINTABLE = database configuration
ACM-DESCRIPTION = The db-config element specifies the data structure that contains
the information needed to connect to a database.
-->

<!--
ACM-ELEMENT = db-host
ACM-PRINTABLE = database hostname
ACM-DESCRIPTION = The db-host element specifies the name of the host computer (the
server) on which the database engine is running.
-->

<!--
ACM-ELEMENT = db-name
ACM-PRINTABLE = database name
ACM-DESCRIPTION = The db-name element specifies the name of the database.
-->

<!--
ACM-ELEMENT = db-user
ACM-PRINTABLE = database user
ACM-DESCRIPTION = The db-user element specifies the user identification used to
connect to the database.
-->

<!--
ACM-ELEMENT = db-password
ACM-PRINTABLE = database password
ACM-DESCRIPTION = The db-password element specifies the password used to connect to
the database.
-->
```

Customize how elements display in the SA Client

In both cases, whether you add these attributes positionally or explicitly, the end result is the same: the value set editor displays the element names (defined in `ACM-PRINTABLE`) and the mouse-over text (defined in `ACM-DESCRIPTION`) in the SA Client, as shown in the following figure.

Custom element names and mouse-over text



	Name	Location	Type	Script
1	hosts_equiv.tpl	/Content/Configurations	Template file	-

Name	Value
⊟ Trusted hosts/users	
⊟ Trusted hosts/users	
Host	
User	

List of hosts and users that are granted "trusted" rexec and rsh access to your system

Filename: /etc/hosts.equiv

Encoding:

Preserve Format:

Preserve Values:

XML configuration template settings

The following table describes all the XML settings available when you create a generic or DTD-based XML configuration template. The list indicates if the setting is required or optional and whether or not it applies only to XML-DTD templates.

XML and XML-DTD template settings

Attribute	Description
ACM-FILENAME-KEY=<key> Required; no default value.	`filename-key` identifies a path to the key in a value set that contains the name of the file being generated.
ACM-FILENAME-DEFAULT=<filename>	`filename-default` identifies the default file name returned if there is no file name in the value set.

XML and XML-DTD template settings, continued

Attribute	Description
Required; no default value.	
ACM-NAMESPACE=<string> Required; no default value.	`namespace` identifies a location where XML elements are stored in the database.
ACM-TIMEOUT=<integer> Optional; (default value is 0)	`timeout` represents the number of minutes that are added to the application configuration's total timeout. A valid timeout is any integer from 0-999 inclusive. The timeouts of all the configuration templates in an application configuration are added together and that number is added to the default timeout of ten minutes for configurations, which is the final timeout value for the entire configuration. Note that any pre- or post-installation scripts in the application configuration that run longer than ten minutes will time out and cancel the entire push job.
ACM-DOCTYPE = <string> Required; no default value. XML-DTD templates only.	`doctype` represents the name of the root element in an XML file. This is in the `DOCTYPE` tag at the beginning of the XML file.
ACM-DOCTYPE-SYSTEM-ID = <string> Required; no default value. XML-DTD templates only.	`system-id` represents the system ID of the DTD file that is the basis of the configuration template. This value is in the DOCTYPE tag at the beginning of the XML file.
ACM-DOCTYPE-PUBLIC-ID = <string> Required; no default value. XML-DTD templates only.	`public-id` represents the public ID of the XML file parsed with the configuration template. This value is in the DOCTYPE tag at the beginning of the XML file DTD options.
ACM-ELEMENT=<element name> Optional XML-DTD templates only.	`element` sets the element that the current options describe. This option defaults to whatever element or attribute comes before this section in the DTD file.
ACM-ATTRIBUTE=<attribute name> Optional XML-DTD templates only.	`attribute` sets the attribute that the current options describe. This option is ignored if no attribute is set. This attribute defaults to whatever element or attribute comes before this section in the file.

XML and XML-DTD template settings, continued

Attribute	Description
ACM-PRINTABLE=<printable> Optional XML-DTD templates only.	`printable` sets the printable value for the element or attribute in the SA Client. This value appears in the value set editor to the left of the field. This is usually set to something short and descriptive.
ACM-DESCRIPTION=<description> Optional XML-DTD templates only.	`description` sets the description for the current element or attribute to be displayed in the SA Client. This value displays when you mouse over the name or value fields in the value set editor. Use this to describe the purpose of the field in the value set editor as well as the valid values for this field.

XML Tutorial 1 - Creating a non-DTD XML configuration template

This tutorial shows how to create a configuration template for a non-DTD XML configuration file. It shows you how to create a configuration template using XML syntax, add it to an application configuration and then attach the application configuration to a managed server. Then you will import values from the `mysql.xml` configuration file on your managed server, make changes to some of those values, and push the new configuration file back to the managed server.

This tutorial is based on the Travel Manager example application described at "Example: Travel manager application and XML configuration file" on page 299.

Sample non-DTD XML mysql.xml file

Below is the contents of the XML configuration file for the travel manager application:

```
<?xml version="1.0" ?>
<db-config>
  <db-host>localhost</db-host>
  <db-name>wrightevents</db-name>
  <db-user>root</db-user>
  <db-password>hp-pass</db-password>
</db-config>
```

1. Creating an XML configuration template

Create a configuration template based on the `mysql.xml` configuration file using the SA Client.

1. From the SA Client navigation pane, select Library and then select the By Type tab.
2. Open the Application Configuration node, then open the Templates node. This shows all the operating system groups.
3. Open an operating system node and select a specific operating system under one of the operating system nodes. For this example, select the operating system of one of your servers where you can install this application configuration.
4. From the **Actions** menu, select **New**.
5. In the Properties view, enter the following information:
   - **Name**: TM-MySql
   - **Description**: This is the template for the mysql.xml configuration file for the Travel Manager application.
   - **Location**: You can leave the default location in the SA Library of /, or select another location to store your template file. Note that the customer setting of the folder containing the template must include the customer setting of the application configuration object. Otherwise the template will not be included in the list of available templates. For more information on folder settings, see the “Folder Permissions” section in the SA 10.50 Administration Guide.
   - **Version**: 0.1.
   - **Type**: Template file
   - **Parser Syntax**: XML Syntax
   - **OS**: Select all the operating systems that the configuration template can be installed on.
   - Select **File > Save**.
6. Keep the Template window open for the next task.

2. Adding XML settings

Since the XML configuration file `mysql.xm`l provides most of the structural settings needed to parse the file’s contents, an XML configuration template in SA only requires three pieces of information in an XML comment: `ACM-NAMESPACE`, `ACM-FILENAME-KEY` and `ACM-FILENAME-DEFAULT`.

1. Select the Content view in the navigation pane.

   ```
   <!--
   ACM-NAMESPACE = /TravelManager
   ACM-FILENAME-KEY = /files/TravelManager
   ACM-FILENAME-DEFAULT = /var/www/html/we/mysql.xml
   -->
   ```

2. Copy and paste the following XML into the Content pane:
3. Select the Validate button to make sure the XML is valid.
4. Select the **File > Save** menu to save your template.
5. Select the **File > Close** menu.

These XML lines define the following:

- `ACM-NAMESPACE`: Specifies a unique name space which is required for each configuration template. In this example, since a name space for the Travel Manager application has already been established, you could reuse the root name space and append the service name. For example:
- `ACM-NAMESPACE = /TravelManager/web/mysql`
- `ACM-FILENAME-KEY`: Specifies a path to the key in the name space that stores the file name of the file being generated.
- `ACM-FILENAME-DEFAULT`: Specifies the path on the target server where the Travel Manger application's `mysql.xml` file is stored. This can be overridden for specific servers or groups of servers.

3. Creating an application configuration to contain the template

In this step, you create an application configuration object to contain your configuration template.

1. In the SA Client navigation pane, select Library and then select the By Type tab.
2. Open the Application Configuration node, then open the Configurations node. This shows all the operating system groups.
3. Open the operating system node and select the same operating system you used when you created the template in the previous steps. The OSs specified for the application configuration must be a subset of the OSs specified for the template.

4. From the **Actions** menu, select **New**.
5. In the Properties view of the File Configuration screen, specify the following properties:
   - **Name**: Tm-MySql-Config
   - **Description**: This is the application configuration for the mySQL configuration file for the Travel Manager application.
   - **Location**: You can leave the default folder location in the SA Library of /, or select another folder to store your application configuration. Note that the customer setting of the folder containing the application configuration must include the customer setting of the managed servers where you intend to push the application configuration. For more information on folder settings, see the “Folder Permissions” section in the SA 10.50 Administration Guide.
   - **Version**: 0.1.
   - **OS**: Select one or more operating systems of managed servers that the application configuration can be installed on.
6. Select **Configuration Values**.
7. Select the add button “+” or the **Actions > Add** menu to add the template.
8. In the Select Configuration Template screen, select the TM-MySql template and then select **OK**. This adds the template to the application configuration object.
9. Select **File > Save** and then **File > Close**. The application configuration and the configuration template inside of it are ready to be attached to a server where the configuration file is stored.

4. Attaching the Application Configuration to a managed server

Now that you have created the configuration template and application configuration object, you need to attach the application configuration to the server where the Travel Manager application is installed and specify the path to where the `mysql.xml` file is stored on the managed server.

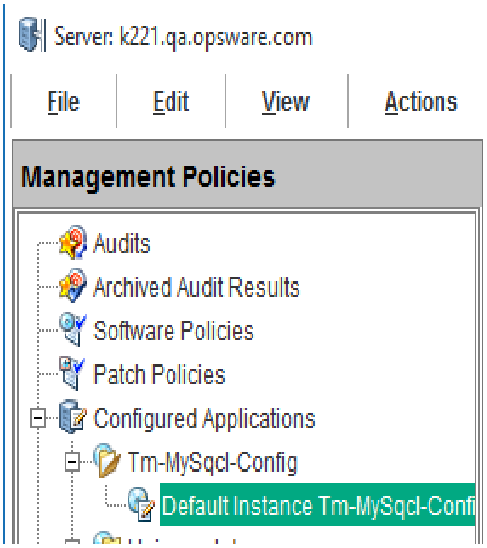
To attach the application configuration to a server:

1. From the SA Client navigation pane, select Devices, then select Servers > All Managed Servers.
2. Locate a server where you can simulate installing the Travel Manager application configuration. The server’s operating system must match one of the operating systems specified in the application configuration.

3. Select the server, and from the **Actions** menu, select **Open**.
4. In the server screen, select the Management Policies tab.
5. In the navigation pane, select Configured Applications. This displays the application configurations that are attached to the server.
6. Select the Installed Configurations tab.
7. From the **Actions** menu, select **Add Configuration**.
8. In the Select Application Configuration screen, select the Tm-MySql-Config application configuration.
9. In the Instance Name field, enter “Default mysql config values”. This creates a value set at the server instance level. For details, see "Value Set Editor at the server level" on page 242.
10. Select **OK**. The application configuration is now attached to the server.
11. Select the Save Changes button. Leave the server screen open for the next step.

The following figure shows the Tm-MySql-Config application configuration attached to the server and the “Default mysql config values” value set. This value set is at the server instance level.

Application Configuration Attached to a Server, with the Value Set at the Server Instance Level Highlighted



> **Note:**
> Note that at this point, if the server you are adding the application configuration to has more than one instance of the `mysql.xml` configuration file because the server is hosting several instances of

> the application, you can right-click the “Default mysql config values” node of the configuration and select **Duplicate**. This creates another value set where you can set the file name path to point to the other instance of the application. For more information on the different levels of value sets, see "Value set levels and value set inheritance" on page 230.

5. Configuring Application Configuration settings for the server

Now that you have attached the application configuration to the managed server, you need to configure it for the server and set values for the configuration file.

To import the values from the configuration file as described below, copy and paste the XML listed in Sample Non-DTD XML File mysql.xml into the target file /var/www/html/we/mysql.xml on your managed server. This will enable the import values step below.

1. Expand the Tm-MySql-Config node to show the value set at the server instance level. This value set is named “Default mysql config values”.
2. From the Contents pane, configure the following settings in the application configuration’s Value Set Editor:
    - **Filename**: The original path and file name of the target XML file on the managed server is displayed to the right of the Filename field. This value is the same value for FILENAME-DEFAULT defined in the template. If this path name is acceptable for this server, you can leave this field empty. If you want the configuration file placed in a different location on this server, set the correct path to the target XML file on the target server in the Filename field.
    - **Encoding**: Select the character encoding for the managed configuration file. The default encoding is the encoding used on the managed server. (Note that UTF-16 encoding is not supported.)
    - **Preserve Format**: Select this option if you want to keep comments and preserve as much of the ordering and spacing of the original XML configuration file from the target server. SA will preserve as much of the target file as possible. For more information, see "Setting fields in the Value Set Editor" on page 233.
    - **Preserve Values**: To preserve the values contained in the actual configuration file on the server when no value is provided in a value set, select **Yes** for this option. With this option set to Yes, the target file’s values will be used unless overridden by values at any level of the inheritance hierarchy. If this option is set to No, and no value exists in the value set, no entry

will be placed in the configuration file. For more information, see "Setting fields in the Value Set Editor" on page 233.

- **Show Inherited Values**: Select this option to show the values in the value set and the inheritance level. When unchecked, only the values set at the current inheritance level are displayed. When checked, all values in the value set are displayed, those set at the current level and those that are inherited. This view is read-only.

3. Right-click inside the value set editor and select **Import Values**. Importing values will read the XML file on the managed server and copy the XML file’s contents to the value set at the server instance level.
4. To save changes, select the Save Changes button. Leave the server screen open for the next step.

6. Editing values and pushing the configuration

The last steps are to edit values in the value set editor and then push the configuration to the server. When you push an application configuration, all the values in the value set replace the values in the configuration files on the target managed servers. Any scripts contained in the application configuration are executed based on what type they are. If the configuration file does not exist on the target server, the file is created when you perform the push.

To edit values and push the application configuration:

1. Modify one or more values of the value set by editing in the Value column. For a description of the columns, see "Columns in the Value Set Editor" on page 234.
2. After you have set values for the application configuration, select **Preview** to see the existing file on the server and the file that would be pushed to the server.
3. Select **Push** to copy the new application configuration to the server.
4. In the Push Configuration screen, select Start Job. Examine the status and results of the push job.

XML Tutorial 2 - Creating an XML-DTD configuration template

This section shows how to create an XML-DTD configuration template to manage an XML configuration file that references a DTD, using the Travel Manager application as an example. For background on the Travel Manager example, see "Example: Travel manager application and XML configuration file" on page 299.

You will first create the XML-DTD template as a text file and then import the text file into the SA Library and add it to an application configuration object. You will then attach the application configuration to a managed server. Finally you will edit some values in the application configuration and push those changes to the target XML file on the managed server.

Sample Travel Manager DTD-based XML file: mysql.xml

For the Travel Manager application, below is the `mysql.xml` XML configuration file. It is stored in `/var/www/html/we/mysql.xml`.

```
<?xml version="1.0"?>
<!DOCTYPE db-config PUBLIC "-//Williams Events//Travel Manager//EN" "mysql.dtd">
<db-config>
       <db-host>localhost</db-host>
       <db-name>wrightevents</db-name>
       <db-user>root</db-user>
       <db-password>hp-pass</db-password>
</db-config>
```

Sample Travel Manager XML DTD file: mysql.dtd

For the Travel Manager application, below is the accompanying `mysql.dtd` DTD. It is stored in `/var/www/html/we/mysql.dtd`.

```
<!ELEMENT db-config (db-host,db-name,db-user,db-password)>
<!ELEMENT db-host     (#PCDATA)>
<!ELEMENT db-name     (#PCDATA)>
<!ELEMENT db-user     (#PCDATA)>
<!ELEMENT db-password (#PCDATA)>
```

1. Creating XML-DTD template in a text editor

In this task, you will create the source for the XML-DTD configuration template using a text editor.

To create an XML-DTD configuration template in a text editor:

1. In a text editor, enter the following information:

   ```
   <!--
   ACM-TIMEOUT = 1
   ACM-FILENAME-KEY = /files/TravelManager
   ACM-FILENAME-DEFAULT = /var/www/html/we/mysql.xml
   ACM-NAMESPACE = /TravelManager/
   ACM-DOCTYPE = db-config
   ACM-DOCTYPE-SYSTEM-ID = mysql.dtd
   ACM-DOCTYPE-PUBLIC-ID = -//Williams Events//Travel Manager//EN
   -->
   ```

   This information is required (except ACM-TIMEOUT) and is used by the application configuration parser to read both the XML-DTD and the XML file you want to manage:

   - `ACM-TIMEOUT`: (Optional) Represents the number of minutes that are added onto the configuration template’s default timeout value (ten minutes) during a push.
   - `ACM-FILENAME-KEY`: Defines the location in name space where the `mysql.xml` filename will stored.
   - `ACM-FILENAME-DEFAULT`: Defines the default location (absolute path) of the `mysql.xml` file on the managed server.
   - `ACM-NAMESPACE`: This value defines the location where values read from the `mysql.xml` file on the managed server will be stored in the database. This name space must be unique, and the path must start with a forward slash (/).
   - `ACM-DOCTYPE`: Defines the name of the root element in the XML file. The root element follows the opening `<!DOCTYPE` declaration found in the target XML configuration file.
   - `ACM-DOCTYPE-SYSTEM-ID`: Defines the name of associated DTD file on the managed server. This value can typically be found in the XML configuration file as the SYSTEM attribute in the DOCTYPE element.
   - `ACM-DOCTYPE-PUBLIC-ID`: Defines a string that represents a public identifier of the XML document. This value can typically be found in the XML configuration file as the PUBLIC-ID attribute of a DOCTYPE element.

2. Save the file, giving it the name `mysql-dtd.tpl`. Keep the file open for the next task.

2. Adding custom settings for element descriptions in the Value Set Editor

In this task you will add some extra information to the XML-DTD template file that allows you to customize the display of each element from the target XML file as seen in the value set editor in the SA Client.

There are two optional settings you can add to your XML-DTD configuration template that allow you to customize how elements from the target XML-DTD configuration file are displayed in the value set editor in the SA Client:

- `ACM-PRINTABLE`: Defines a label for each element from the XML file that will be displayed in the value set editor when the XML-DTD template is shown in the SA Client.
- `ACM-DESCRIPTION`: Defines mouse-over text when a user moves a mouse pointer over the field defined in `ACM-PRINTABLE` in the value set editor in the SA Client.

See the XML-DTD template figure for an example of how these elements appear in the value set editor in the SA Client.

This example uses the explicit method for placing these custom settings inside the XML-DTD template. For more information on this method of placing custom settings, see "Explicit versus positional display settings" on page 304.

To add custom settings in the XML-DTD template:

1. In the text editor with the `mysql-dtd.tpl` file, add the following information for each XML element being referenced by the DTD. For example, after the main information in the template, add a list of each element contained in the source XML file and then for each element, make an XML comment using these three ACM setting tags:
   - `ACM-ELEMENT`: Declares the element from the XML file that the following the `ACM-PRINTABLE` and `ACM-DESCRIPTION` settings will describe. This option defaults to whatever element or attribute came before this section in the DTD file.
   - `ACM-PRINTABLE`: Set a short, descriptive label for the element when it is displayed in the value set editor.
   - `ACM-DESCRIPTION`: Set mouse-over text for the element.

   The example XML-DTD template file should look like this:

```
ACM-TIMEOUT = 1
ACM-FILENAME-KEY = /files/TravelManager
ACM-FILENAME-DEFAULT = /var/www/html/we/mysql.xml
ACM-NAMESPACE = /TravelManager/
ACM-DOCTYPE = db-config
ACM-DOCTYPE-SYSTEM-ID = mysql.dtd
ACM-DOCTYPE-PUBLIC-ID = -//Williams Events//Travel Manager//EN
-->
<!ELEMENT db-config (db-host,db-name,db-user,db-password)>
<!ELEMENT db-host     (#PCDATA)>
<!ELEMENT db-name     (#PCDATA)>
<!ELEMENT db-user     (#PCDATA)>
<!ELEMENT db-password (#PCDATA)>
<!--
ACM-ELEMENT = db-config
ACM-PRINTABLE = database configuration
ACM-DESCRIPTION = The db-config element specifies the data structure that
contains the information needed to connect to a database.
-->
<!--
ACM-ELEMENT = db-host
ACM-PRINTABLE = database hostname
ACM-DESCRIPTION = The db-host element specifies the name of the host computer
(the server) on which the database engine is running.
-->
<!--
ACM-ELEMENT = db-name
ACM-PRINTABLE = database name
ACM-DESCRIPTION = The db-name element specifies the name of the database.
-->
<!--

ACM-ELEMENT = db-user
ACM-PRINTABLE = database user
ACM-DESCRIPTION = The db-user element specifies the user identification used
to connect to the database.
-->
<!--
ACM-ELEMENT = db-password
ACM-PRINTABLE = database password
ACM-DESCRIPTION = The db-password element specifies the password used to
connect to the database.
-->
```

2. Save and close the file.

3. Importing the XML-DTD configuration file

In this task you will import your template file and create a new configuration template that will manage the target XML and DTD files.

To import the XML-DTD configuration file into the SA Library:

1. From the SA Client navigation pane, select Library and then select the By Type tab.
2. Locate and open the Application Configuration node. Open the Templates node. Open an operating system group and navigate to the operating system that the template file applies to. Note that a template can apply to multiple operating systems. For example, you could select one of the Red Hat operating system versions.
3. From the **Actions** menu, select **Import Template**.
4. Navigate to the file you created in the previous step and select it. Set the encoding if it is not the default.
5. Select Open. This imports your template file and displays it in the Templates screen.
6. Select the Properties view and enter the following information:
   - **Name:** mysql-dtd.tpl
   - **Description**: This is the template for the mysql.dtd (mysql.xml) file for the Travel Manager application.
   - **Location**: **Specify where in the SA Library you want to store the template.**
   - **Version**: 0.1.
   - **Type**: Template file
   - **Parser Syntax**: XML DTD Syntax.
   - **OS**: Select the appropriate operating system.
7. Select the Contents view to display the contents of the template file you just imported.
8. Select the **Validate** button to confirm that the syntax is valid before proceeding.
9. Select **File** > **Save**.
10. Select **File** > **Close.**

4. Creating an Application Configuration object

An application configuration is a container for configuration template files. In this step you will create an application configuration and import the template.

1. From the SA Client navigation pane, select Library and then select the By Type tab.
2. Locate and open the Application Configuration node. Open the Configurations node. Open the operating system group and select the operating system that the application configuration applies to. Note that an Application Configuration can apply to multiple operating systems. You can change this in a later step.
3. Select the **Actions** > **New** menu. This displays the File Configuration screen where you can specify the properties and contents of the Application Configuration.
4. In the Properties view, specify the following:
    - **Name: TM-mysql-dtd**
    - **Description**: This is the application configuration for the mysql.xml and mysql.dtd files for the Travel Manager application.
    - **Location: Specify where in the SA Library you want to store the Application Configuration.**
    - **Version**: 0.1
    - **OS**: Select the appropriate operating system.
5. In the Content view, select the **Actions** > **Add** menu or the “+” button to add a template to the Application Configuration.
6. In the Select Configuration Template screen, select the mysql-dtd.tpl template file.
7. Select **OK**.
8. Select **File** > **Save** to save your Application Configuration.
9. Select **File** > **Close.**

5. Attaching the Application Configuration to a managed server

In this task you will to attach the application configuration to the server where the Travel Manager application is installed, and then enter the path name to the `mysql.dtd` configuration file.

To attach the application configuration to a server:

1. From the SA Client navigation pane, select Devices > Servers All Managed Servers.
2. Select a server, and from the **Actions** menu, select **Open**. Make sure the operating system of the server you select matches the operating system specified on the application configuration and the template.
3. Select the Management Policies tab.
4. Select the Configured Applications node.
5. Select the Installed Configurations tab.
6. From the **Actions** menu, select **Add Configuration**.
7. In the Select Application Configuration screen, select the application configuration TM-mysql-dtd.
8. In the Instance Name field, enter “Value set 1 for mysql.xml”.
9. Select **OK**. The application configuration is attached to the server.
10. Select the **Save Changes** button.
11. Leave the application configuration screen open for the next step.

6. Importing values from the configuration file

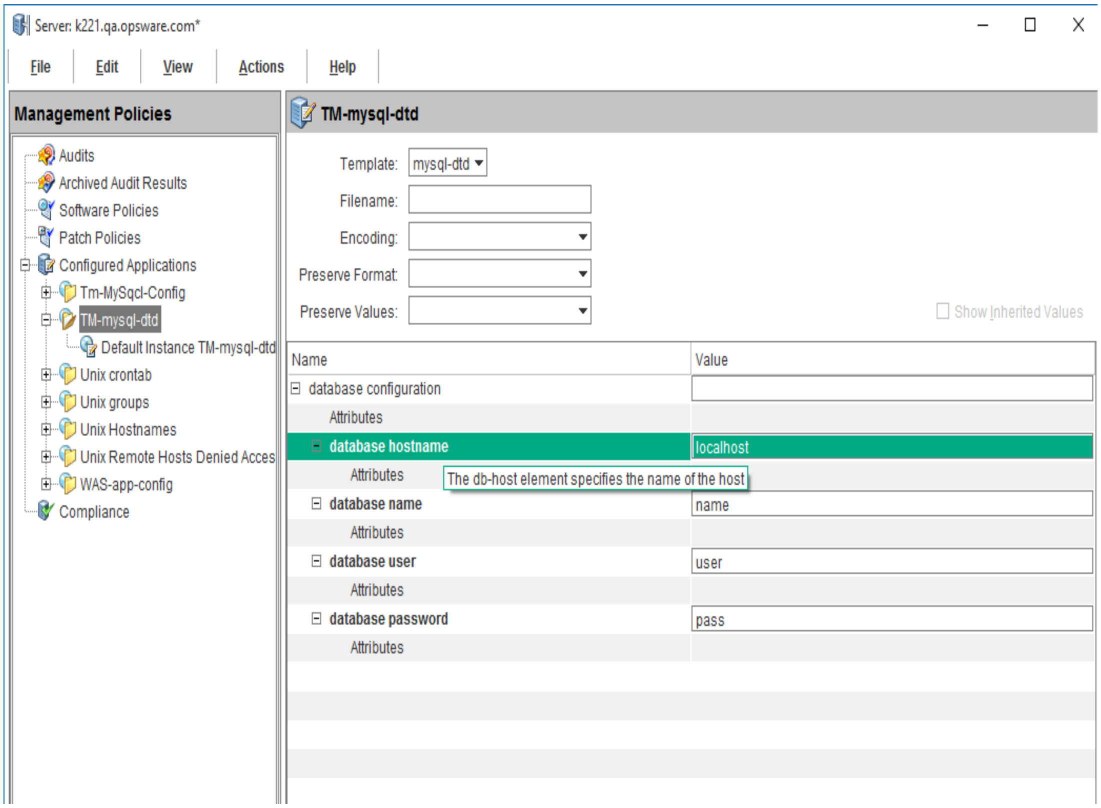
The next step is to set values in a value set. While you can set values in a value set manually, the easiest way is to import the values from an existing configuration file. In this step you will import the values from a configuration file on the server.

To import the values from the configuration file as described below, copy and paste the XML listed in Sample Travel Manager DTD-based XML File: mysql.xml above into the target file /var/www/html/we/mysql.xml on your managed server. Copy and paste the DTD listed in Sample

Travel Manager XML DTD File: mysql.dtd above into the target file `/var/www/html/we/mysql.dtd`. This will enable the import step below.

1. In the server Management Policies, open the Configured Applications node. This displays the application configuration attached to the server.
2. Open the TM-mysql-dtd node. This displays the server instance value sets, which are the node under the TM-mysql-dtd node.
3. Select the “Value set 1 for mysql.xml” node. This is the server instance value set for the mysql-dtd.tpl configuration template.
4. Right-click on any value under the Value column and select the **Import Values** menu.
5. In the Confirmation Dialog, select Yes. This imports the values from the file /var/www/html/we/mysql.xml into the value set at the server instance level.
6. Select the Save Changes button. The following figure shows the XML-DTD template with the server instance value set and the mouse-over text displayed from the ACM-DESCRIPTION element.

Value Set and Mouse-Over Text for XML-DTD Configuration Template



Template: mysql-dtd
Filename:
Encoding:
Preserve Format:
Preserve Values:

Name	Value
⊟ database configuration	
Attributes	
⊟ database hostname	localhost
Attributes	
⊟ database name	name
Attributes	
⊟ database user	user
Attributes	
⊟ database password	pass
Attributes	

The db-host element specifies the name of the host

7. Leave the application configuration displayed for the next step.

7. Editing values and push the configuration

The last step is to edit values in the value set editor and then push the configuration to the server. When you push an application configuration, all the values in the value set replace the values in the configuration files on the target managed servers. All scripts in the application configuration are also executed. If the configuration file does not exist on the target server, the file is created as part of the push.

To edit values and push the application configuration:

1. Make sure that the server instance level value set is displayed by selecting the “value set 1 for mysql.xml” node in the navigation pane.
2. Modify the password value under the Value column.
3. Select **Save Changes**.
4. Select **Preview** to display the difference between the existing configuration file on the server and the configuration file that will be pushed to the server.
5. After examining the comparison screen, select the Close button.
6. Select **Push**. This displays the Push Configurations wizard.
7. Select **Start Job**. This starts the push operation.
8. Examine the Job Status of the push job. Select any step to display details on that step.
9. When the job completes, select the Close button.
10. You can log on to the server and examine the `mysql.xml` configuration file to verify that it was updated on the server.

CML Tutorial 1 - Creating an Application Configuration for a simple web app server

This section demonstrates how to set up and manage a simple configuration file for a Web Application Server running on two servers. Each server runs the Web Application Server and needs to be configured separately. This tutorial shows how to create an application configuration, a configuration template, value sets and two instances of the application configuration, one for each server. Finally it shows how to push the application configuration to each server.

1. Determining the configuration files to be managed

The web application server uses one configuration file named WASconfig.txt. This file is located in the directory /opt/WAS/WASconfig.txt. The contents of this file are as follows:

```
size=1000
dir=/tmp/WAS_001
primary=yes
```

2. Creating a template for the configuration file

You can create a template in either of two ways:

- Create a template in a text file and import the text file into the SA Library.
- Create a template directly in the SA Library.

Both methods are described below. Choose one method and follow the steps.

Creating a template file and importing it into SA

1. In a text editor, copy the configuration file into an empty file:
   ```
   size=1000
   dir=/tmp/WAS_001
   primary=yes
   ```
2. Create a template that models this file using CML. First add a comment block and the required CML metadata defining the name space and the target configuration file name.
   - The name space defines the key where information for this template will be stored in the database.
   - The file name key defines where the default file name will be stored in the database.
   - The default file name specifies the name that will be used for the resulting configuration file.

     ```
     @#####################################################
     # /opt/WAS/WASconfig.txt                             #
     # Version 1.0                                        #
     # Author <name>                                      #
     ######################################################@
     @!namespace=/WAS-server-namespace/@
     ```

```
@!filename-key=”/WAS-server”@
@!filename-default=”/opt/WAS/WASconfig.txt”@
size=1000
dir=/tmp/WAS_01
primary=yes
```

3. Next change the variable parts of the configuration file to variables using CML tags:

```
@#####################################################
# /opt/WAS/WASconfig.txt                            #
# Version 1.0                                       #
# Author <name>                                     #
#####################################################@
@!namespace=/WAS-server-namespace/@
@!filename-key=”/WAS-server”@
@!filename-default=”/opt/WAS/WASconfig.txt”@
size=@value_of_size;int@
dir=@value_of_dir;string@
primary=@value_of_primary;boolean@
```

4. Save the file with an extension of “.tpl”, for example `WASconfig_txt.tpl`.
5. Import the template file into the SA Library. Follow the steps at "Importing and validating a template file" on page 277.

Creating a template file directly in SA

1. In the SA Client, select the **Library** tab.
2. Select the **By Type** tab.
3. Open the Application Configurations node and the Templates node. Navigate to the OS family and the OS version where the application runs. For this example, select Red Hat Enterprise Linux AS 4.
4. Select **Actions > New**. This displays the Templates screen.
5. Enter the template name “WASconfig_txt.tpl” and a brief description. Select the location in the SA Library to store the template file. Set the version string. Set the Type to “Template file”. Set the Parser Syntax to “CML Syntax”.
6. Select the **Content** view to display a text editor.
7. Type or paste in the CML text. This is the same CML text as shown above.
8. Select **Actions > Validate** to check the syntax of your CML. Make any needed corrections.

9. Select **File > Save** to save your template.
10. Close the template screen.

3. Creating an Application Configuration object

Create an application configuration object to contain the configuration template.

1. In the SA Client, select the **Library** tab.
2. Select the **By Type** tab.
3. Open the Application Configurations node and the Configurations node. Navigate to the OS family and the OS version where the application runs. For this example, select Red Hat Enterprise Linux AS 4.
4. Select **Actions** > **New**. This displays the Configuration screen.
5. Enter the name “WAS-app-config”, a brief description and version string of your application configuration. Select the location in the SA Library to store the application configuration.
6. Select **File** > **Save** to save your application configuration.

4. Adding the template file to the Application Configuration object

1. Open the “WAS-app-config” application configuration object you created in the previous step.
2. Select the **Configuration Values** view.
3. Select the “+” button or select **Actions** > **Add**. This displays the Select Configuration Template screen.
4. Select your “**WASconfig_txt.tpl**” template file and select **OK**.
5. Select **File** > **Save** to save your changes to the application configuration object.
6. Select **File** > **Close** to close the application configuration object.

5. Attaching the Application Configuration object to servers

Two servers are running the web application server, RHEL001 and RHEL008. RHEL001 is the primary server and RHEL008 is the secondary server. Create two instances of the application configuration by attaching the application configuration object to these two servers as follows:

1. Locate the primary server RHEL001 in the SA Client.
2. Select the RHEL001 server and select **Actions** > **Open**.
3. Select the Management Policies tab.
4. Select the Configured Applications node.
5. Select the **Actions** > **Add Configuration** menu.
6. Select the “**WAS-app-config**” application configuration object.
7. In the Instance Name field, enter “Primary Instance of WAS-app-config” and select **OK**.
8. Select **Save Changes**. This creates an instance of the application configuration for the server RHEL001.
9. Repeat the above steps with the secondary server RHEL008 except in the Instance Name field, enter “Secondary Instance of WAS-app-config” and select OK. This creates a second instance of the application configuration for the server RHEL008.

6. Setting default values

The required values for the configuration files for the two servers are shown below:

Configuration value for two servers running the Web Application Server

Server: RHEL001	Server: RHEL008

`size=1000`
`dir=/tmp/WAS_001`
`primary=yes`
	`size=1000`
	`dir=/tmp/WAS_008`
	`primary=no`

You can set default values for the configuration file that can be inherited by the individual servers or that can be overridden by each individual server. If an individual server does not override the default value, it uses the inherited default value.

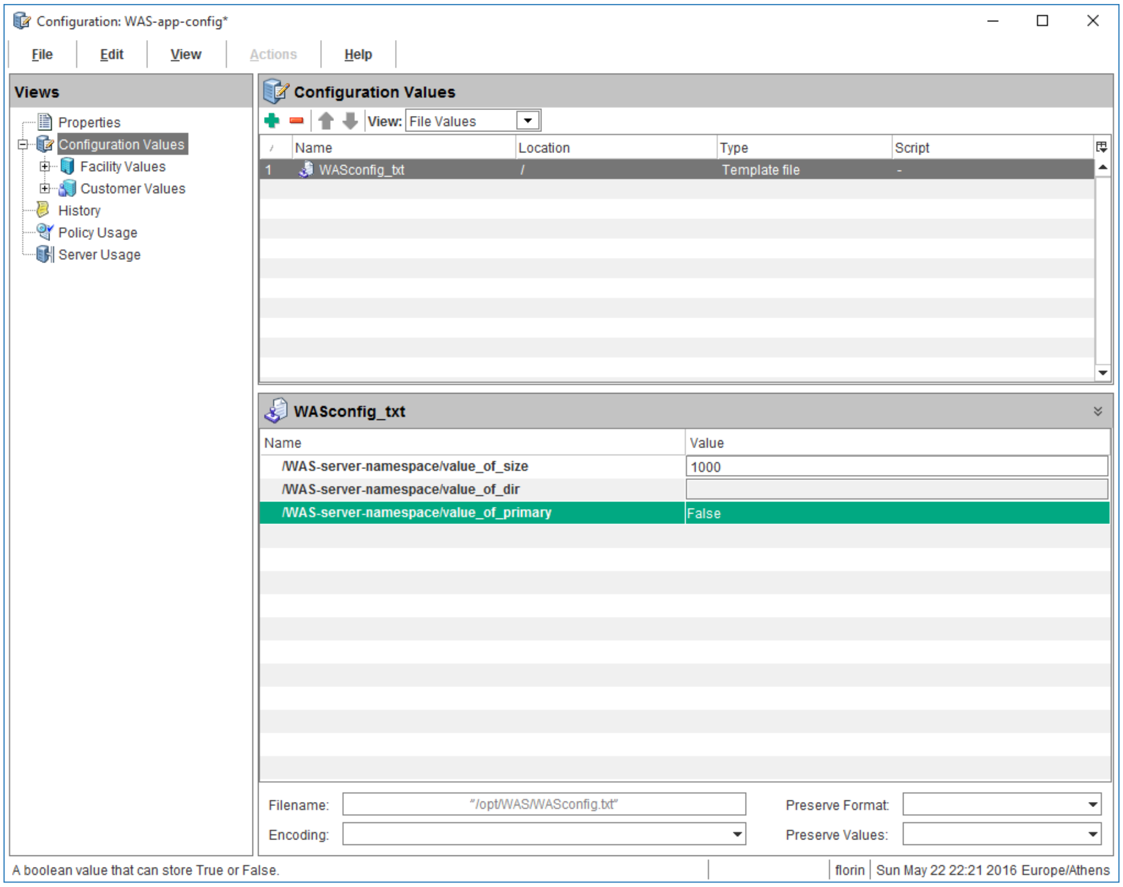
The following table shows which values will be set with default values and which will be set by individual servers:

Table 7: Application level default vValues for the configuration file for the Web Application Server

Default Value	Description
`size=1000`	Set this to a default value of 1000 at the application level. All servers attached to this application configuration will use this value unless they override it.
`dir`	Do not set this to a default value. Each server will set this value at the server level or at the server instance level.
`primary=no`	Set this to a default value of “no” at the application level. All servers attached to this application configuration will use this value unless they override it.

Setting application level default values

1. Open the application configuration object you created above.
2. Select the **Configuration Values** view.
3. Select the **WAS-config-template.tpl** template file.
4. Select **File Values** in the view drop-down list. This displays the default values for the template at the application level.
5. Set “value_of_size” to 1000 and “value_of_primary” to “False” (case matters), as shown below. Do not set a default value for “value_of_dir” because each server will need to set this value.



	Name	Location	Type	Script
1	WASconfig_txt	/	Template file	-

Name	Value
/WAS-server-namespace/value_of_size	1000
/WAS-server-namespace/value_of_dir	
/WAS-server-namespace/value_of_primary	False

Filename: "/opt/WAS/WASconfig.txt"

Encoding:

Preserve Format:

Preserve Values:

A boolean value that can store True or False.

florin | Sun May 22 22:21 2016 Europe/Athens

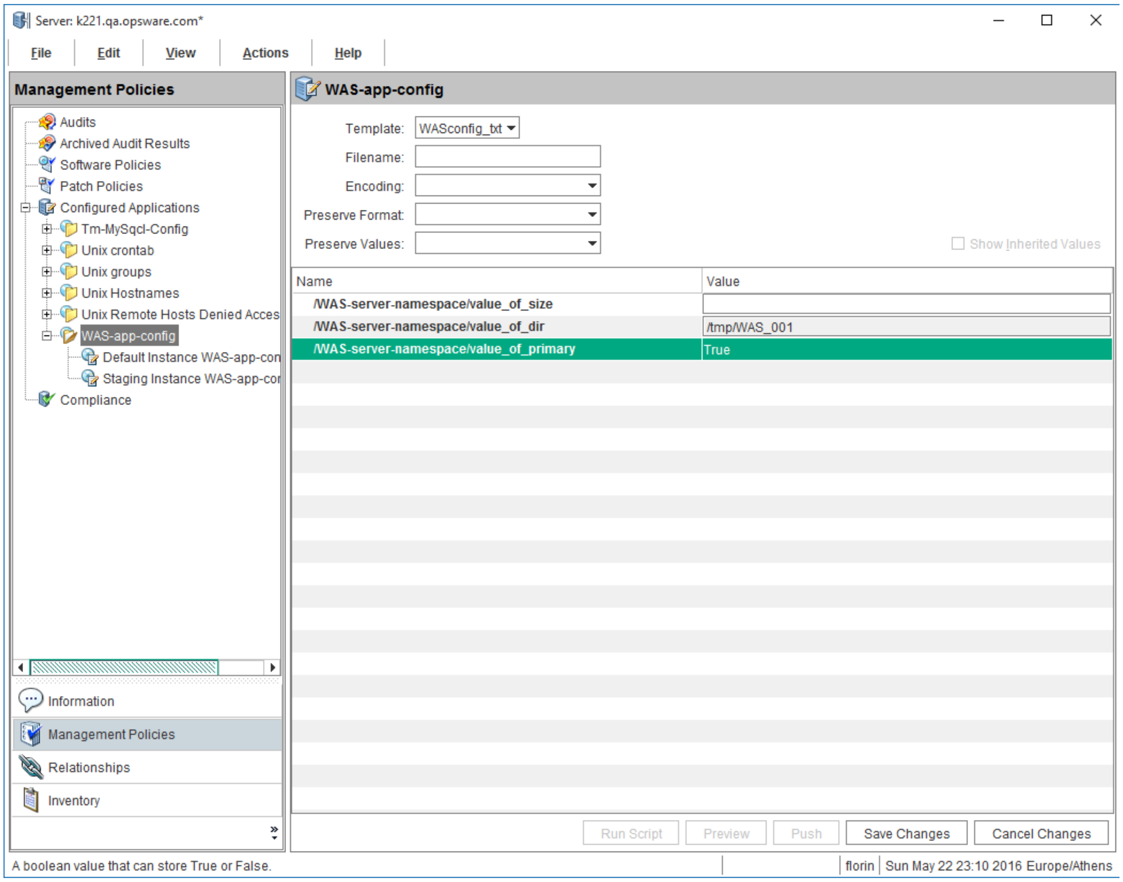
6. Select **File** > **Save** to save your application level default values.
7. Select **File** > **Close**.

Setting server level default values for RHEL001

The server RHEL001 needs to set `dir=/tmp/WAS_001` and `primary=yes` at the server level. It does not need to set size because it can use the value set at the application level.

1. Locate in the SA Client the RHEL001 server.
2. Select the RHEL001 server and select Actions > Open.
3. Select the Management Policies tab.
4. Open the Configured Applications node to reveal the “WAS-app-config” application configuration object.
5. Select the “WAS-app-config” application configuration object attached to this server. This displays the default values set at the server level. Values set at the server level apply to all instances of the application configuration on the server unless overridden at the server instance level.

6. Set the server level default value for “value_of_dir” to “/tmp/WAS_001”, as shown below. Do not set a default value for “value_of_size” or “value_of_primary” because these values will be inherited from the application level.



Template: WASconfig_txt
Filename:
Encoding:
Preserve Format:
Preserve Values:

Name	Value
/WAS-server-namespace/value_of_size	
/WAS-server-namespace/value_of_dir	/tmp/WAS_001
/WAS-server-namespace/value_of_primary	True

A boolean value that can store True or False. | florin | Sun May 22 23:10 2016 Europe/Athens

7. Select the **Save Changes** button or the File > Save menu to save your **server level default values**.

8. Open the WAS-app-config node to reveal the application configuration instance “Primary Instance of WAS-app-config”.

9. Select the instance “Primary Instance of WAS-app-config”. This displays the **instance level default values**. This is the lowest value set level and overrides all other levels. Notice that no values have been defined at the instance level.

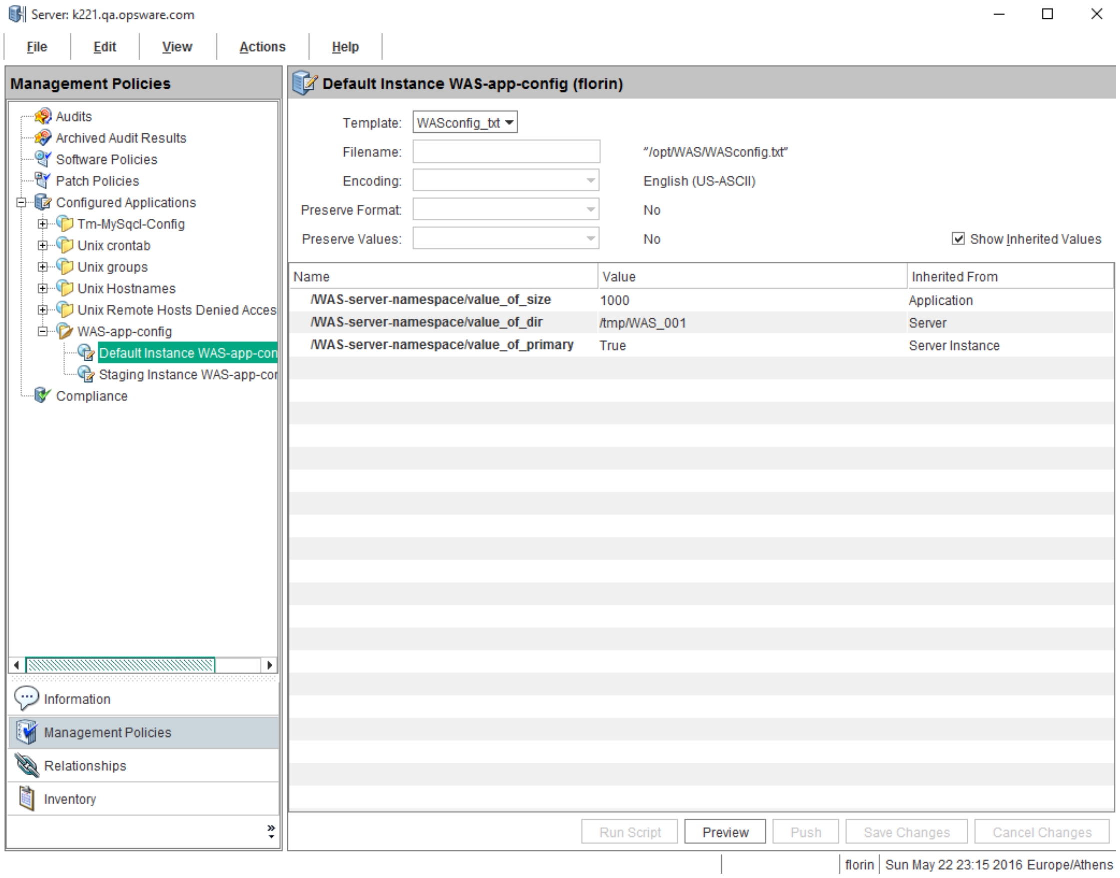
10. Select “Show Inherited Values” to show the values that will be inherited from the application level defaults and the server level defaults. Notice that “value_of_size” and “value_of_primary” are inherited from the application level and “value_of_dir” is inherited from the server level.

11. Uncheck “Show Inherited Values” so you can set the instance level default values.

12. Set “value_of_primary” to “True” (case matters).

13. Select the **Save Changes** button or the File > Save menu to save your **instance level default values**.

14. Select “Show Inherited Values” again to show the values that are inherited from the application level, the server level and the instance level. Notice that “value_of_size” is inherited from the application level, “value_of_dir” is inherited from the server level and “value_of_primary” is inherited from the instance level, as shown below.



Default Instance WAS-app-config (florin)

Template: WASconfig_txt

Filename: "/opt/WAS/WASconfig.txt"

Encoding: English (US-ASCII)

Preserve Format: No

Preserve Values: No

☑ Show Inherited Values

Name	Value	Inherited From
/WAS-server-namespace/value_of_size	1000	Application
/WAS-server-namespace/value_of_dir	/tmp/WAS_001	Server
/WAS-server-namespace/value_of_primary	True	Server Instance

florin | Sun May 22 23:15 2016 Europe/Athens

Setting server level default values for RHEL008

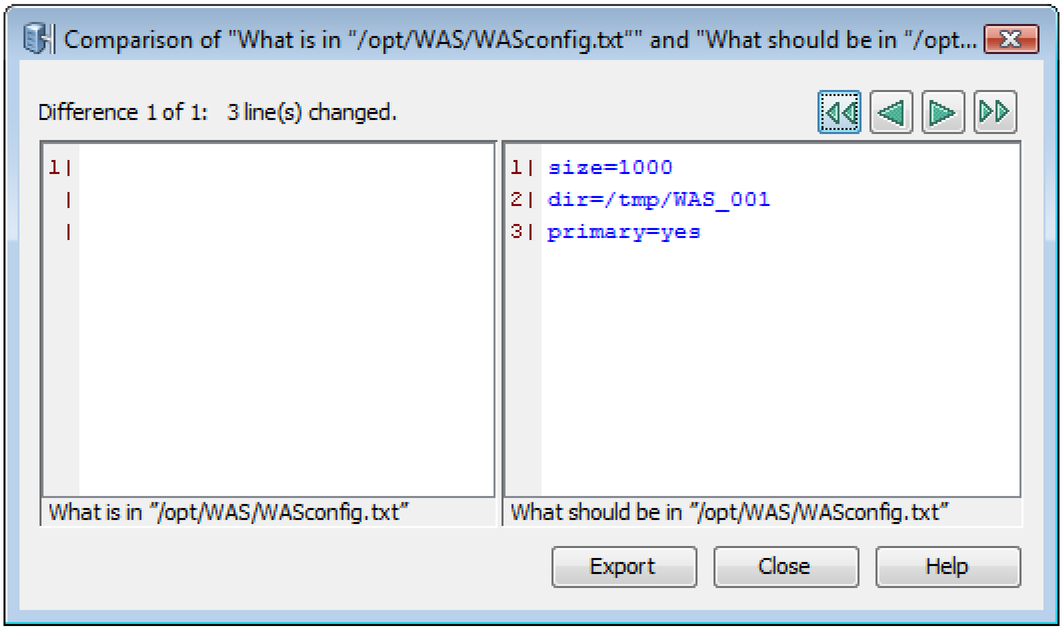
The server RHEL008 needs to set `dir=/tmp/WAS_008` at the server level. It does not need to set size or primary because it can use the values set at the application level.

1. Locate in the SA Client the RHEL008 server.
2. Select the RHEL008 server and select **Actions > Open**.
3. Select the **Management Policies** tab.

4. Open the Configured Applications node to reveal the “WAS-app-config” application configuration object.
5. Select the “WAS-app-config” application configuration object attached to this server. This displays the default values set at the server level. Values set at the server level apply to all instances of the application configuration on the server unless overridden at the server instance level.
6. Set the server level default value for “value_of_dir” to “/tmp/WAS_008”. Do not set a default value for “value_of_size” or “value_of_primary” because these values will be inherited from the application level.
7. Select **Save Changes** or the **File > Save** menu to save your **server level default values**.
8. Open the WAS-app-config node to reveal the application configuration instance “Secondary Instance of WAS-app-config”.
9. Select the instance “Secondary Instance of WAS-app-config”. This displays the **instance level default values**. This is the lowest value set level and overrides all other levels. Notice that no values have been defined at the instance level.
10. Select “Show Inherited Values” to show the values that will be inherited from the application level defaults and the server level defaults. Notice that “value_of_size” and “value_of_primary” are inherited from the application level and “value_of_dir” is inherited from the server level.

7. Comparing the actual configuration files with the configuration template

You can optionally compare the values specified in the application configuration to the actual values in the configuration file on the server by selecting the Preview button from the server screen. The following shows the comparison on RHEL001 when there is no configuration file on the server yet:


Comparison of "What is in "/opt/WAS/WASconfig.txt"" and "What should be in "/opt...

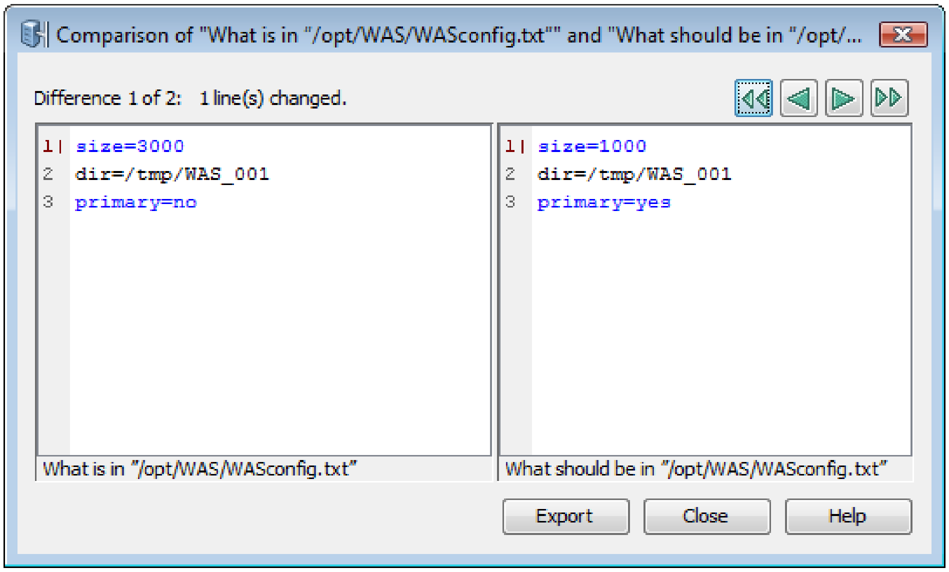
Difference 1 of 1: 3 line(s) changed.

What is in "/opt/WAS/WASconfig.txt"

What should be in "/opt/WAS/WASconfig.txt"

```
size=1000
dir=/tmp/WAS_001
primary=yes
```

The following shows the comparison when there is an existing configuration file on the server that differs from the values specified in the application configuration:


Comparison of "What is in "/opt/WAS/WASconfig.txt"" and "What should be in "/opt/...

Difference 1 of 2: 1 line(s) changed.

What is in "/opt/WAS/WASconfig.txt"

```
size=3000
dir=/tmp/WAS_001
primary=no
```

What should be in "/opt/WAS/WASconfig.txt"

```
size=1000
dir=/tmp/WAS_001
primary=yes
```

8. Pushing configuration changes to the server

1. Locate in the SA Client the RHEL001 server.
2. Select the RHEL001 server and select **Actions** > **Open**.

3. Select the **Management Policies** tab.
4. Open the Configured Applications node to reveal the “WAS-app-config” application configuration object.
5. Open the “**WAS-app-config**” application configuration node to reveal the “Primary Instance of WAS-app-config” instance.
6. Select the “**Primary Instance of WAS-appconfig**” instance.
7. Select the **Push button**.
8. You can select the **Start Job** button to accept the job defaults for scheduling and notifications, or select **Next.**
9. In the Scheduling screen you can specify when you want the push configurations job to run. Select **Next**.
10. In the Notifications screen you can specify one or more people to receive an email message when the job succeeds or fails. You can also specify a ticket identifier. Select Next.
11. Select **Start Job**. SA generates the configuration file from the template and value set, pushes the resulting configuration file to the server and displays the results.
12. Select **Close**.

For a tutorial showing a more complex configuration file, see "CML Tutorial 2 - Creating a template of a web server configuration file" below.

CML Tutorial 2 - Creating a template of a web server configuration file

This tutorial explains how to use the **Configuration Modeling Language** (CML) to make a configuration template based upon the Microsoft Internet Information Services (IIS) Web server configuration file named UrlScan.ini. You will use the CML language to create a template file based on this file so it can be managed on a managed server.

While this tutorial will not teach you everything about CML, creating a CML template from UrlScan.ini will help you gain a fundamental understanding of CML and the process of creating a configuration template from a configuration file.

A sample of the UrlScan.ini file is listed in Sample UrlScan.ini File. The complete CML file is listed in Complete url_scan_ini.tpl CML Template.

To complete this tutorial you should have the following.

- Documentation for `UrlScan.ini`. This is available with the Microsoft IIS documentation.
- The `UrlScan.ini` file.
- A text editor for creating a CML file.

1. Analyzing the native configuration file and documentation

Once you have identified an application configuration file you want to manage, the first thing to do is to analyze the native configuration file and its documentation. Make sure that you understand the purpose of the configuration file, all the elements in the file and the kinds of data the configuration file manages.

This tutorial uses the UrlScan.ini file. A sample is listed in Sample UrlScan.ini File. The file UrlScan.ini enables systems administrators to configure Microsoft Internet Information Services (IIS) web server. The UrlScan.ini file consists of several sections, such as [Options], [AllowVerbs], [DenyVerbs], [DenyHeaders], [AllowExtensions], and [DenyExtensions]. Each section allows the IIS administrator to set different configurations to either allow or disallow certain kinds of HTTP requests on the IIS server. These sections do not need to be in any specific order. However, the information inside each section must be ordered. For example, the [AllowVerbs] section must be followed by specific HTTP requests that are allowed to access the web site.

UrlScan.ini contains lists of strings, such as lists of verbs and file extensions, and options that take a Boolean value of “1” for True or “2” for False.

2. Creating a CML comment block

A CML template is a simple text file named with the .tpl file extension. Use a text editor to create a new text file named Url_Scan_ini.tpl. The .tpl extension is the typical (but optional) file extension used by SA for CML templates.

Create a CML comment block at the top of the file with information about the template as follows:

```
@#############################################
# \system32\inetsrv\urlscan.ini (Windows)    #
# Version 1.0                                #
# Joe Author (joe_author@your_company.com)   #
##############################################@
```

The CML comment tag uses the following syntax:

```
@# <one line comment>
```

Or

```
@## <comments spanning multiple lines>
        <comments spanning multiple lines> #@
```

3. Creating CML setup instructions

The setup section instructs the parser how to interpret the CML file. The `namespace`, `filename-key` and `filename-default` instructions are required for all CML files. The other instructions shown below are optional. These instructions define whitespace handling, Boolean values, comment formats and ordering rules.

To create the basic setup section, enter the following information after the comment block in the CML template:

```
@!namespace=/security/@
@!filename-key="/test";filename-default="/c/UrlScan.ini"@
@!optional-whitespace@
@!boolean-yes-format="1";boolean-no-format="0"@
@!line-comment-is-semicolon@
@!unordered-lines@
```

Notice that each CML instruction tag starts with the characters “`@!`” and ends with the character “`@`”.

The following line combines two CML instructions on one line:

```
@!filename-key="/test";filename-default="/c/UrlScan.ini"@
```

This line could also be written as two separate lines as follows:

```
@!filename-key="/test"@
@!filename-default="/c/UrlScan.ini"@
```

Setup instructions

The following figure explains setup instructions.

CML template setup instructions

CML Tag	Description
@!namespace=/security/@	The `@!namespace` instruction defines the name space that will be used by this CML template. This defines where in the database the values used by this CML template will be stored. Each CML template should use its own name space so names do not collide with other

CML template setup instructions, continued

CML Tag	Description
	template. In this example, the name space is `/security.` All value sets will be stored in this name space.
@!filename-key="/files/urlscan_ini";filename-default="/c/urlscan.ini"@	The `@!filename-key` instruction defines the location in the name space where the file name will stored. If the value starts with a “/”, it defines a separate name space. If the value does not start with a “/”, it will be appended to the name space defined by the `@!namespace` instruction. In this example, the default file name will be stored in the database under the name space “`/file/urlscan-ini`”. The `@!filename-default` instruction defines the default path where the native configuration file will be saved on the server. This path can be changed using the SA Client. In this example, when the configuration file is pushed to a managed server, it will be placed in `/c/urlscan.ini`. Note that the path names use only forward slashes.

@!optional-whitespace@	This instruction indicates that whitespace is optional between items in the configuration file. For example, either of the following entries would be valid if this option is set:
	Key = "value"
	`Key="value"`

@!boolean-yes-format="1";boolean-no-format="0"@	This instruction defines the allowable Boolean values in the configuration file. In this case, true is indicated with the character `1`, and false is indicated with a `0`. Any other values for Booleans will not be allowed.
@!line-comment-is-semicolon@	Instructs the parser to ignore anything that follows a semicolon in the configuration file. This allows comments in the native configuration file using the semicolon before each comment.
@!unordered-lines@	Instructs the parser that the sections in the configuration file can be in any order. If you used `ordered-lines`, then the configuration file would have to conform to the order of the template.

4. Defining the [Options] section — Opening blocks

Now you are ready to add CML instructions to the template. The first section of the UrlScan.ini file you will model in CML is the [Options] section, which contains several options for the configuration file.

In CML, if a section of information in a configuration file has more than one kind of data (data that needs to be read differently by the CML parser), you can open “blocks” to handle each section of information separately. Typically, you open a block in CML to define special parser rules for a section of the CML file. The [Options] section has two “blocks” of information: the title of the section which is just “[Options]” and all the options in that section. Since these blocks belong together, you will set them at different levels, the first block (the title of the section) at level one, and the second block (the contents of the section) at level two. Nesting the blocks in this manner keeps the sections within the block together when read by the parser.

1. To define the [Options] section, enter the following lines:

   ```
   @1[;optional;ordered-lines@
   [Options]
   @2[;unordered-lines@
   ```

2. In the UrlScan.ini file the [Options] section contains a list of key-value pairs. Use the block tag (`[`) set at two levels because there are two kinds of data in this section: a heading and a list of key-value pairs. The first level block handles the text string “[Options]” while the second level block handles all of the key-value pairs in that section.

The following table explains how to open two block levels for the [Options] section.

Marking up the start of the [Options] section

CML tag	Description

@1 [;optional;ordered-lines@	The number 1 sets the first level of the multiline block.
	`[`
	The square bracket opens a new block.
	`optional`
	Indicates that this entire block is optional in the configuration file.
	`ordered-lines`
	Indicates that whatever follows this tag (the string [Options]) must come first in the native UrlScan.ini configuration file. In other words, the title [Options] must appear before the actual options.

[Options]	The string that names the section in the native configuration file. This string will

Marking up the start of the [Options] section, continued

CML tag	Description
	appear in the configuration file.

@2[;unordered-lines@	The number 2 sets the second level of the block.
	`[`
	The square bracket opens a new block, in this case a level 2 block, nested within the previous level 1 block.
	`unordered lines`
	Indicates that the lines that follow [Options] within the block can be in any order in the configuration file. That is, all the key-value pairs in the [Options] section can be in any order.

Next, you will define all the options that can appear in the [Options] section of the configuration file. Most of these entries use the CML replace tag, because they are simple key-value pairs that allow you to replace a single value. The following table explains the CML for each option.

Table 10: Marked up key-value pairs from UrlScan.ini [Options] section

CML Tag	Description

AllowDotInPath = @allow_dot_in_path;boolean@	Note: All of the key-value pairs use some variation of the following syntax (unless otherwise indicated):
	`string literal = @source;type@`
	The string literal defines the actual option name that will appear in the configuration file. The source is the location in the database where the value will be stored in the value set. The type is the type of data that will be stored in the value set.
	`@allow_dot_in_path`
	This string defines the name space path in which to store this value. In this example, the name space is relative, which means that it will be appended to the name space that you defined in the header of the template (@!namespace=/security/@) and will store the value in that name space location. That is, the value will be stored in the database at the key `/security/allow_dot_in_path.`
	You could also write this tag as follows:
	`AllowDotInPath = @/security/allow_dot_in_path;boolean@`
	boolean
	Since the key-value pair type is Boolean, the CML type: `boolean` is used. Note that since the header of this template defined the Boolean true value as 1, when the IIS administrator sets the value set, they would need to enter a 1 to allow dots in the path of the IIS server.

AllowHighBitCharacters =	Similar to the previous example, AllowHighBitCharacters is the option

Table 10: Marked up key-value pairs from UrlScan.ini [Options] section, continued

CML Tag	Description
@allow_high_bit_ characters;boolean@	that appears in the configuration file, allow_high_bit_characters is the relative name space path, and boolean is the data type. This IIS option allows users to choose whether or not high bit characters are acceptable in a URL, indicated by a 1 for true and 0 for false in the configuration file.
AllowLateScanning = @allow_late_ scanning;boolean@	Allows the IIS administrator to choose whether or not late scanning of a URL is acceptable. Defines a name space location to store the value. `boolean` indicates this key can be 1 for true and 0 for false in the configuration file.
AlternateServerName = @alternate_servername@	Defines a name space location where an alternate server name can be stored when entered by the user or read in from a configuration file. Since no type is specified, the default is a string type.
EnableLogging = @enable_ logging;boolean@	Allows users to turn on logging, indicated by 1 for true and 0 for false in the configuration file.
LoggingDirectory = @logging_directory;dir@	Allows users to choose a directory to store log files, if logging has been turned on. The type `dir` indicates a directory.
LogLongURLs = @log_ long_urls;boolean@	Allows users to choose whether or not to log URLs that access the server, specified by 1 for true and 0 for false in the configuration file.
NormalizeUrlBeforeScan = @normalize_url_before_ scan;boolean@	Allows users to choose whether or not to normalize the URL before it is read by the server, indicated by 1 for true and 0 for false in the configuration file.
PerDayLogging = @per_ day_logging;boolean@	Allows users to choose to turn on per day logging, indicated by 1 for true and 0 for false in the configuration file.
PerProcessLogging = @per_process_ logging;boolean@	Allows users to turn on or off per process logging, indicated by 1 for true and 0 for false in the configuration file.

RejectResponseUrl =
@reject_response_ url;string;r”(HTTP_ URLSCAN_STATUS_ HEADER)|(HTTP_ URLSCAN_ORIGINAL_ VERB)|(HTTP_URLSCAN_ ORIGINAL_ URL)”;optional@
	Syntax:
	string literal = @source;type;r”regular expression”;option@
	`reject response`
	String literal that defines the path where the strings will be stored in the name space.
	`string`
	Indicates that the data type for the reject URL request is a string.
	`r”`
	This is a string range specifier that introduces a regular expression. In this case, a range of string literals.

Table 10: Marked up key-value pairs from UrlScan.ini [Options] section, continued

CML Tag	Description

	`(HTTP_URLSCAN_STATUS_HEADER)|(HTTP_URLSCAN_ORIGINAL_VERB)|(HTTP_URLSCAN_ORIGINAL_URL)”`
	The string literals (rejected URL responses) to be read by the parser: the status header, original verb, and original URL.
	`optional`
	Indicates that this value is optional. That is, the RejectResponseUrl option may be omitted from the UrlScan.ini file.

RemoveServerHeader = @remove_server_header;boolean@	Allows users to turn on or off the RemoveServerHeading feature. When activated (set to 1), the reject response sent to the client will remove the server header in the message. This setting is indicated by 1 for true and 0 for false in the configuration file.
UseAllowVerbs = @use_allow_verbs;boolean@	Allows users to turn on or off the UseAllowVerbs feature. When activated (set to 1), the server will reject any request to the server that contains an HTTP verb that is not explicitly listed in the AllowVerbs section of the UrlScan.ini file. Indicated by 1 for true and 0 for false in the configuration file.
UseAllowExtensions = @use_allow_extensions;boolean@	Allows users to turn on or off the UseAllowExtension feature. When activated (set to 1), the server will reject any request to the server that contains a file extension that is not explicitly listed in the AllowExtension section of the UrlScan.ini file. Indicated by 1 for true and 0 for false in the configuration file.
UseFastPathReject = @use_fast_path_reject;boolean@	Allows users to turn on or off the UseFastPathReject feature. When activated (set to 1), the server ignores the RejectResponseUrl option and returns a short 404 response to the client when a URL is rejected. Indicated by 1 for true and 0 for false in the configuration file.
VerifyNormalization = @verify_normalization;boolean@	Allows users to turn on or off normalization of all URLs scanned by UrlScan.ini. When activated (set to 1), the URL is normalized before being scanned. Indicated by 1 for true and 0 for false in the configuration file.

5. Defining the [AllowExtensions] section - Closing a block by opening a new block

Now that you have defined all of the options in the [Options] section of the UrlScan.ini file, you are ready to start defining the next section, [AllowExtensions]. Remember that to start the [Options] section you had to open a two-level block to account for two levels of information — the title of the [Options] section and its contents.

Before you can start defining the [AllowExtensions] section, you need to close the previous section by closing the CML block. With CML, you can close a block explicitly with the "]" tag, or by opening a new block at a higher level (specified by a lower number) or at an equal level. In this task, you will open the new block for the [AllowExtensions] the same way you opened a block for the [Options] section, by starting a new level 1 block. This automatically closes the blocks opened by the [Options] section.

To open a new block and define the [AllowExtensions] section:

1. After the last line of the [Options] section, enter the following to open the new block for the [AllowExtensions] section:

   ```
   @1[;optional;ordered-lines@
   [AllowExtensions]
   @2[;unordered-lines@
   ```

   The following table explains how opening a new two level block closes the previous block.

   Starting a New Block for the [AllowExtensions] Section

CML Tag	Description

@1 [;optional;ordered-lines@	The number 1 opens a new level one block. Because it is a number 1 level block, which is at a higher level than the previous block (a level two block for the key-value pairs in the [Options] section) and equal to the level 1 block before that, it will close the two blocks that came before it.
	Note that you can explicitly close a block by using the close block tag. For example:
	`@2]@`
	`[`
	CML block tag that opens a new block.
	`optional`
	Indicates that this entire block is optional and not required to be in the configuration file.
	`ordered-lines`
	Indicates that whatever follows this tag (the string [AllowExtensions] has to come first in the native UrlScan.ini configuration file. In other words, you could not list all the options in the native file and then the title. [AllowExtensions] has to come first. In CML, the ordered-line element determines this order.

[AllowExtensions]	The literal string that names the section in the native configuration file.

Starting a New Block for the [AllowExtensions] Section, continued

CML Tag	Description
@2[;unordered-lines@	The number 2 sets the second level of the block. [CML block symbol that opens a new block. `unordered lines` Indicates that all the lines that follow [AllowExtensions] within the block can be in any order in the configuration file. That is, all the key-value pairs in the [AllowExtensions] section can be in any order.

2. Next, because the [AllowExtensions] section of the UrlScan.ini file can contain any list of file extensions entered by the user, you will use a CML loop and loop target tag to instruct the parser to read the information in this section one line at a time. Immediately after the last `@2 [;unordered-lines@` text from the last step, enter the following text:

```
@*allow_extension;unordered-string-set@
.@.@
```

The following table explains the how the loop and loop target CML tags work:

Loop and loop target CML tags

CML tag	Description

@*allow_extension;unordered-string-set@	Syntax
	@<level><tag type><name>;<data type>;<options>@
	The loop tag (`*`) will loop and read over the unordered string set listed in the [AllowExtensions] section.
	`allow_extension`
	String that defines the path where the strings will be stored in the name space.
	`unordered-string-set`
	Indicates that the list of strings do not have to be in any specific order.

.@.@	First (`.`) In this section, this unordered string set that the parser reads is a list of file extensions listed in the [AllowExtensions] section that start with a (`.`) character. @.@ Loop target tag (`.`) instructs the parser to read everything in this list that starts with a period character.

3. Save the file.

6. Defining the [DenyExtensions] section

Next you define the [DenyExtensions] section of the UrlScan.ini file the same way you defined the [AllowExtensions] section. You will open a new level one block, which closes the previous block from the [AllowExtensions] section. Then you will open a level two block from which you will instruct the parser to read an unordered list of all file extensions beginning with a (.) that you want to block using UrlScan.ini.

The CML for the [DenyExtensions] section looks like this:

```
@1[;optional;ordered-lines@
[DenyExtensions]
@2[;unordered-lines@
@*deny_extension;unordered-string-set@
.@.@
```

7. Defining the [AllowVerbs] and [DenyVerbs] sections

The next two sections of the UrlScan.ini file follow the same CML you used for [DenyExtensions] in the previous sections. You open a first level block to close the previous block, which will also parse the following text as an ordered line.

Then you open a second level block that reads the following list of unordered strings — in other words, a list of verbs. In these two sections, the string instructs the parser to read the list of verbs you want to allow into your web site and a list of verbs you want to deny access to your web site.

The CML for both of these sections is as follows:

```
@1[;optional;ordered-lines@
[AllowVerbs]
@2[;unordered-lines@
@*allow_verb;unordered-string-set@
@.@

@1[;optional;ordered-lines@
[DenyVerbs]
```

```
@2[;unordered-lines@
@*deny_verb;unordered-string-set@
@.@
```

8. Defining the [DenyHeaders] section

Next you define the [DenyHeaders] section of the UrlScan.ini file, which allows you to configure IIS to deny specific HTTP request headers.

This section is similar to the previous sections in that you open two blocks for strings. However, you will separate the list of HTTP headers listed in the UrlScan.ini file by a colon, using a CML sequence delimiter. Since HTTP request headers contain a colon (:), you need to use a sequence delimiter to tell the parser to read each line in the section so when it encounters a colon (:), it will move on to the next entry.

For example, the list of HTTP headers to be denied listed in the UrlScan.ini file might read:

Translate:
If:
Lock-Token:

Because each header request listed in the configuration file ends with a (:), you need to instruct the parser to recognize the (:) as the end of an entry.

1. To define the [DenyHeaders] section, after the last line of the [DenyVerbs] section, enter the following text to open the new block for the [DenyHeaders] section:

   @1[;optional;ordered-lines@
   [DenyHeaders]
   @2[;unordered-lines@

   As in previous sections, these tags open a level one block to be read as an ordered line, then they open a second level block to be read as unordered lines.

2. Next enter the following CML loop and loop target tags to instruct the parser to read through the list of header requests:

   ```
   @*deny_header;unordered-string-set;;sequence-delimiter=":"@
   @.@:
   ```

The following table describes the syntax of these two tags.

Loop and Loop Target Tags for the [DenyHeaders] Section

CML tag	Description

@*deny_header;unordered-string-set;;sequence-delimiter=":"@	`*`
	Indicates a loop CML tag that will read through the list of strings.
	`deny_header`
	String literal that defines the path where the strings will be stored in name spacename space.
	`unordered-string-set`
	Indicates that the list of strings can be listed in any order.
	`;`
	The first semicolon separates the two sections of the tag.
	`;`
	The second semicolon allows you to enter the following colon (:) sequence delimeter without it being interpreted as a range.
	`sequence-delimiter=":"`
	Instructs the parser to read a colon (:) as part of the string and the point at which to move on to the next entry.

@.@	Loop target tag instructs the parser to store these values into the `deny_header` name space location. For example: /security/deny_header.
:	The final colon (:) tells the parser that each item in this list will be followed by a colon. That is, this character will be stored as a part of the entry for a denied header.

3. Save the file.

9. Defining the [DenyURLSequences] section

Define the [DenyUrlSequence] similar to the [DenyHeader] section. Open two blocks that will be read for order and unordered strings. However, for this section you will separate the list of URL sequences in the template with a field delimiter. The field delimiter used here will be an end of line element which instructs the parser to stop reading an entry when it encounters the end of a line.

To define the [DenyUrlSequence] section:

1. After the last line of the [DenyUrlSequence] section, enter the following text to open the new block for the [DenyUrlSequence] section:

   ```
   @1[;optional;ordered-lines@
   [DenyUrlSequence]
   @2[;unordered-lines@
   ```

   As in previous sections, these tags open a level one block to be read as an ordered line, then they open a second level block to be read as unordered lines.

2. Next type the following CML loop and loop target tags to instruct the parser to read through the list of URL sequences to be denied:

   ```
   @*deny_url_sequence;unordered-string-set;;field-delimiter-is-eol@
   @.@
   ```

   The following table describes the syntax of these tags.

   TLoop and Loop Target Tags for the [DenyUrlSequence] Section

CML Tag	Description

@*deny_url_ sequence;unordered-string-set;;field-delimiter-is-eol@	`*`
	Indicates a loop CML tag that reads through the list of strings.
	`deny_url_sequence`
	String literal that defines the path where the string will be stored in the name space.
	`unordered-string-set`
	Indicates that the list of strings can be listed in any order.
	`;`
	The first semicolon separates the two sections of the tag.
	`;`
	The second semicolon allows you to enter the following field delimeter without it being interpreted as a range.
	`field-delimiter-is-eol`
	Instructs the parser to read the next entry up to the end of the line.

@.@	Loop target tag instructs the parser to store these values into the `deny_url_sequence` name space location. For example: /security/deny_url_sequence.

3. Save the file.

10. Defining the [RequestLimits] section

Defining the [RequestsLimits] is very similar to the way you defined the [DenyUrlSequence] section. Open two blocks that will be read for order and unordered strings. But for this section, after you open both blocks, you will use the CML replace tag to define three key-value pairs.

To define the [RequestsLimits] section:

1. After the last line of the [RequestsLimits] section, enter the following text to open the new block for the [RequestsLimits] section:

   ```
   @1[;optional;ordered-lines@
   [RequestsLimits]
   @2[;unordered-lines@
   ```

   As in previous sections, these tags open a level one block to be read as an ordered line, then open a second level block to be read as unordered lines. Recall that by starting the new first level block, you are closing the previous second level block from the {DenyUrlSequence] section.

2. Next, type the following `CML replace` tags to define the three key-value pairs found in the [RequestsLimits] section:

   ```
   MaxAllowedContentLength = @max_allowed_content_length;int@
   MaxUrl = @max_url;int@
   MaxQueryString = @max_query_string;int@
   @1]@
   ```

   The following table describes the syntax of these tags.

   Loop and loop target tags for the [DenyUrlSequence] section

CML Tag	Description

MaxAllowedContentLength = @max_allowed_content_length;int@	`MaxAllowedContentLength`
	Request limit parameter string from the configuration file.
	`max_allowed_content_length`
	String literal that defines the path where the value will be stored in the name space.
	`int`
	Indicates that the value to be stored is an integer.

Loop and loop target tags for the [DenyUrlSequence] section, continued

CML Tag	Description

MaxUrl = @max_url;int@	`MaxUrl`
	Request limit parameter string from the configuration file.
	`max_url`
	String literal that defines the path where the value will be stored in the name space.
	`int`
	Indicates that the value to be stored is an integer.

MaxQueryString = @max_query_string;int@	`MaxQueryString`
	Request limit parameter string from the configuration file.
	`max_query_string`
	String literal that defines the path where the value will be stored in the name space.
	`int`
	Indicates that the value to be stored is an integer.

@1]@	This level one block tag closes the block.

3. Save the file.

11. Placing the template in an Application Configuration

Now that you have created the CML template for UrlScan.ini and saved it as url_scan_ini.tpl, you are ready to do the following tasks:

- Import the template into the SA Client and validate the CML syntax. See "Importing and validating a template file" on page 277.
- Add the template to an Application Configuration. See " Adding or removing templates from an Application Configuration" on page 279.
- Attach the Application Configuration to a server. See " Attaching an Application Configuration to a server or device group" on page 283.

- Test your template by making changes and pushing them to a server. See "Pushing Application Configurations" on page 287.

These steps are described in "CML Tutorial 1 - Creating an Application Configuration for a simple web app server" on page 324.

Sample UrlScan.ini File

Below is a sample UrlScan.ini file.

```
[Options]

UseAllowVerbs=1                ; If 1, use [AllowVerbs] section, else use the

                               ; [DenyVerbs] section.   The default is 1.


UseAllowExtensions=0           ; If 1, use [AllowExtensions] section, else

                               ; use the [DenyExtensions] section. The

                               ; default is 0.


NormalizeUrlBeforeScan=1       ; If 1, canonicalize URL before processing.

                               ; The default is 1. Note that setting this

                               ; to 0 will make checks based on extensions,

                               ; and the URL unreliable and is therefore not

                               ; recommend other than for testing.


VerifyNormalization=1          ; If 1, canonicalize URL twice and reject

                               ; request if a change occurs. The default

                               ; is 1.


AllowHighBitCharacters=0       ; If 1, allow high bit (ie. UTF8 or MBCS)

                               ; characters in URL. The default is 0.


AllowDotInPath=0               ; If 1, allow dots that are not file

                               ; extensions. The default is 0. Note that
                               ; setting this property to 1 will make checks
                               ; based on extensions unreliable and is
                               ; therefore not recommended other than for
                               ; testing.

RemoveServerHeader=1           ; If 1, remove the 'Server' header from
                               ; response. The default is 0.

EnableLogging=1                ; If 1, log UrlScan activity. The
                               ; default is 1. Changes to this property
                               ; will not take effect until UrlScan is
                               ; restarted.

PerProcessLogging=0            ; This property is deprecated for UrlScan
                               ; 3.0 and later. UrlScan 3.0 and later can
                               ; safely log output from multiple processes
                               ; to the same log file. Changes to this
                               ; property will not take effect until
                               ; UrlScan is restarted.

AllowLateScanning=0            ; If 1, then UrlScan will load as a low
                               ; priority filter. The default is 0. Note
                               ; that this setting should only be used in
                               ; the case where there another installed
                               ; filter is modifying the URL and you wish
                               ; to have UrlScan apply its rules to the
                               ; rewritten URL. Changes to this property
                               ; will not take effect until UrlScan is
                               ; restarted.

PerDayLogging=1                ; If 1, UrlScan will produce a new log each
```

```
                               ; day with activity in the form
                               ; 'UrlScan.010101.log'. If 0, UrlScan will
                               ; log activity to urlscan.log. The default
                               ; is 1. Changes to this setting will not
                               ; take effect until UrlScan is restarted.

UseFastPathReject=0            ; If 1, then UrlScan will not use the
                               ; RejectResponseUrl. On IIS versions less
                               ; than 6.0, this will also prevent IIS
                               ; from writing rejected requests to the
                               ; W3SVC log. UrlScan will log rejected
                               ; requests regardless of this setting. The
                               ; default is 0.

LogLongUrls=0                  ; This property is deprecated for UrlScan 3.0
                               ; and later. UrlScan 3.0 and later will
                               ; always include the complete URL in its log
                               ; file.

UnescapeQueryString=1          ; If 1, UrlScan will perform two passes on
                               ; each query string scan, once with the raw
                               ; query string and once after unescaping it.
                               ; If 0, UrlScan will only look at the raw
                               ; query string as sent by the client. The
                               ; default is 1. Note that if this property is
                               ; set to 0, then checks based on the query
                               ; string will be unreliable.

RejectResponseUrl=

LoggingDirectory=Logs
```

```
[AllowVerbs]

;
; The verbs (aka HTTP methods) listed here are those commonly
; processed by a typical IIS server.
;
; Note that these entries are effective if "UseAllowVerbs=1"
; is set in the [Options] section above.
;

GET
HEAD
POST

[DenyVerbs]

;
; The verbs (aka HTTP methods) listed here are used for publishing
; content to an IIS server via WebDAV.
;
; Note that these entries are effective if "UseAllowVerbs=0"
; is set in the [Options] section above.
;

PROPFIND
PROPPATCH
MKCOL
DELETE
PUT
COPY
```

```
MOVE
LOCK
UNLOCK
OPTIONS
SEARCH

[DenyHeaders]

;
; The following request headers alter processing of a
; request by causing the server to process the request
; as if it were intended to be a WebDAV request, instead
; of a request to retrieve a resource.
;

Translate:
If:
Lock-Token:
Transfer-Encoding:

[AllowExtensions]

;
; Extensions listed here are commonly used on a typical IIS server.
;
; Note that these entries are effective if "UseAllowExtensions=1"
; is set in the [Options] section above.
;

.htm
.html
```

```
.txt
.png
.png
.png

[DenyExtensions]

;
; Extensions listed here either run code directly on the server,
; are processed as scripts, or are static files that are
; generally not intended to be served out.
;
; Note that these entries are effective if "UseAllowExtensions=0"
; is set in the [Options] section above.
;
; Also note that ASP scripts are denied with the below
; settings. If you wish to enable ASP, remove the
; following extensions from this list:
;    .asp
;    .cer
;    .cdx
;    .asa
;

; Deny executables that could run on the server
.exe
.bat
.cmd
.com

; Deny infrequently used scripts
```

```
.htw     ; Maps to webhits.dll, part of Index Server
.ida     ; Maps to idq.dll, part of Index Server
.idq     ; Maps to idq.dll, part of Index Server
.htr     ; Maps to ism.dll, a legacy administrative tool
.idc     ; Maps to httpodbc.dll, a legacy database access tool
.shtm    ; Maps to ssinc.dll, for Server Side Includes
.shtml   ; Maps to ssinc.dll, for Server Side Includes
.stm     ; Maps to ssinc.dll, for Server Side Includes
.printer ; Maps to msw3prt.dll, for Internet Printing Services

; Deny various static files
.ini     ; Configuration files
.log     ; Log files
.pol     ; Policy files
.dat     ; Configuration files
.config  ; Configuration files

[DenyUrlSequences]
;
; If any character sequences listed here appear in the URL for
; any request, that request will be rejected.
;

..  ; Don't allow directory traversals
./  ; Don't allow trailing dot on a directory name
\   ; Don't allow backslashes in URL
:   ; Don't allow alternate stream access
%   ; Don't allow escaping after normalization
&   ; Don't allow multiple CGI processes to run on a single request
```

Complete url_scan_ini.tpl CML template

Below is the complete url_Scan_ini.tpl template.

```
@#############################################
# \system32\inetsrv\urlscan.ini (Windows)    #
# Version 1.0                                #
# Joe Author (joe_author@your_company.com)   #
##############################################@

@!namespace=/security/@
@!filename-key="/test";filename-default="/c/UrlScan.ini"@
@!optional-whitespace@
@!boolean-yes-format="1";boolean-no-format="0"@
@!line-comment-is-semicolon@
@!unordered-lines@

@###########################################
# Begin data                               #
###########################################@

@1[;optional;ordered-lines@
[Options]
@2[;unordered-lines@
AllowDotInPath = @allow_dot_in_path;boolean@
AllowHighBitCharacters = @allow_high_bit_characters;boolean@
AllowLateScanning = @allow_late_scanning;boolean@
AlternateServerName = @alternate_servername@
EnableLogging = @enable_logging;boolean@
LoggingDirectory = @logging_directory;dir@
LogLongURLs = @log_long_urls;boolean@
NormalizeUrlBeforeScan = @normalize_url_before_scan;boolean@
PerDayLogging = @per_day_logging;boolean@
PerProcessLogging = @per_process_logging;boolean@
RejectResponseUrl =
@reject_response_url;string;r”(HTTP_URLSCAN_STATUS_HEADER)|(HTTP_URLSCAN
_ORIGINAL_VERB)|(HTTP_URLSCAN_ORIGINAL_URL)”;optional@
RemoveServerHeader = @remove_server_header;boolean@
UnescapeQueryString = @unescape_query_string;boolean@
UseAllowVerbs = @use_allow_verbs;boolean@
UseAllowExtensions = @use_allow_extensions;boolean@
UseFastPathReject = @use_fast_path_reject;boolean@
VerifyNormalization = @verify_normalization;boolean@

@1[;optional;ordered-lines@
[AllowExtensions]
```

```
@2[;unordered-lines@
@*allow_extension;unordered-string-set@
.@.@

@1[;optional;ordered-lines@
[DenyExtensions]
@2[;unordered-lines@
@*deny_extension;unordered-string-set@
.@.@

@1[;optional;ordered-lines@
[AllowVerbs]
@2[;unordered-lines@
@*allow_verb;unordered-string-set@
@.@

@1[;optional;ordered-lines@
[DenyVerbs]
@2[;unordered-lines@
@*deny_verb;unordered-string-set@
@.@

@1[;optional;ordered-lines@
[DenyHeaders]
@2[;unordered-lines@
@*deny_header;unordered-string-set;;sequence-delimiter=":"@
@.@:

@1[;optional;ordered-lines@
[DenyURLSequences]
@2[;unordered-lines@
@*deny_url_sequence;unordered-string-set;;field-delimiter-is-eol@
@.@

@1[;optional;ordered-lines@
[RequestLimits]
@2[;unordered-lines@
MaxAllowedContentLength = @max_allowed_content_length;int@
MaxUrl = @max_url;int@
MaxQueryString = @max_query_string;int@
@1]@
```

CML primer

This section introduces the **Configuration Modeling Language**, CML. For complete details on CML see "CML Reference" on page 373. See also "CML Tutorial 1 - Creating an Application Configuration for a simple web app server" on page 324 and "CML Tutorial 2 - Creating a template of a web server configuration file" on page 335.

SA manages configuration files by creating a **configuration template** that it uses to:

- Model the syntax of the configuration file.
- Extract the values from the configuration file and store them as a **value set** in the SA database. Once those values are stored, you can use the SA Client to manage those values.
- Create a new configuration file from the value set.
- Push the new configuration file to your servers.
- Audit the configuration files on your servers to ensure compliance.

Terminology

- **Configuration File** - The file to be managed by SA.
- **Value Set** - The data values from configuration files that can vary from server to server. Values in a value set are stored in the SA database in “key = value” format.
- **Name Space** - The structure of how value sets are stored in the SA database.
- **Configuration Template** - A model of your configuration file written in CML.
- **Configuration Modeling Language (CML)** - The set of instruction tags that are used to model a configuration file in a configuration template.
- **Instruction** or **Tag** - The key words and characters that define the action to be taken. All instructions start and end with the “@” character. The terms “instruction” and “tag” can be used interchangeably.
- **Application Configuration Object** - A container of configuration templates that, when combined with a value set, generates a configuration file that is then “pushed” to your managed servers. This can also contain scripts that are executed as part of the push operation.

CML basic concepts

CML, Configuration Modeling Language, models the syntax of a configuration file. You use CML to create a **configuration template**, which is a model of the target configuration file. To do this, it is usually best to obtain the documentation for the target configuration file so you can understand the valid values and ranges in the configuration file and determine the best way to model it.

Configuration templates work best when the entire configuration file is modeled by CML. However, it is possible to only write CML for some of the lines in a configuration file. This is called a partial template and is discussed at "Partial templates" on page 373.

Required CML instruction tags

The following three CML instructions (also called CML tags) are required in any configuration template.

- `namespace` defines the key where the value set data is stored in the SA database. See "Namespace tag" below.
- `filename-key` defines the key where the target configuration file name is stored in the SA database. See "Filename-key tag" on the next page.
- `filename-default` specifies the default name of the target configuration file. See "Filename-default tag" on page 363.

All other tags are optional and are used to model the contents of the specific configuration file. For complete details on CML, see "CML Reference" on page 373. For details on these three required tags, see "CML global option attributes" on page 400.

Namespace tag

The `namespace` instruction defines the key where value sets are stored in the SA database.

Syntax:

```
@!namespace=<path>@
```

where <*path*> is a string similar to a directory path that defines the key where value sets are stored in the SA database.

Example:

```
@!namespace=/example/namespace/@
```

This example sets the base name space for all other instructions in this template to “/example/namespace/”. That is, all values in the value set will be stored at the key “/example/namespace/” unless specified otherwise.

Description:

The namespace instruction specifies the key in the key/value mapping for the values in the value set. It is an arbitrary string and can be anything except a number. This instruction determines the key where the values in the value set are stored in the SA database. The Replace tag (and other tags) in the configuration file use the name space key to obtain values from the SA database.

The namespace value must be absolute. Subsequent values can have relative or absolute path names.

- Absolute names are the full path name starting with a “/” character. These names do not use the value specified in the namespace instruction. For example, any value matching the following tag:

  ```
  @/testval@
  ```

  would get stored in the value set under the key “/testval”.

- Relative names get appended to the value specified in the namespace instruction tag. For example, any value matching the following tag:

  ```
  @testval@
  ```

  would get stored in the value set under the key “/example/namespace/testval”.

Note that any named tag that is part of a loop requires a dot “.” in front of the name, and that tags name space gets appended to the current loop's name space. For example: `@.testval@`

Filename-key tag

The filename-key instruction specifies the key where the target configuration file name gets stored in the SA database.

Syntax:

```
@!filename-key=<path>@’
```

where <`path`> is an arbitrary string similar to a directory path that defines the key where configuration file name is stored in the SA database.

Examples:

```
@!filename-key=/files/example@
```

This example specifies that the file name will be stored in the SA database with the key “/files/example”. Note that because the <path> value starts with a “/” character, it is an absolute path.

```
@!filename-key=files/example@
```

This example specifies a relative path because the value does not start with a “/” character. If it were combined with the previous namespace example, the configuration file name would be stored at /example/namespace/files/example”.

Description:

The filename-key instruction specifies the key that will be used to store the target configuration file name in the SA database. For example, when you set the “Filename” field for a template in the SA Client, that file name will be stored under the key “/files/example” in the value set. Note that the filename-key is not a file system path, but just a key that can be written similar to a path.

This can be very handy for pre and post scripts that need to know the name of the configuration file before or after it has been pushed to the target servers. For example, if you have a post script that needs to add a line to the end of the configuration file after it has been pushed, it might look something like this:

```
echo “#end of the file” >> @/files/example@
```

Filename-default tag

Syntax:

```
@!filename-default=<file>@
```

where <`file`> is the directory path and file name of the target configuration file in the file system of your managed servers. This is where the generated configuration file will be pushed on to your managed servers.

Example:

```
@!filename-default=/etc/hosts@
```

This example specifies that the target configuration file is /etc/hosts. This value is stored in the SA database with the key specified in the Filename-key instruction.

Description:

The filename-default instruction specifies the standard file system path of the target configuration file. This value is the default file name and directory and is stored under the key specified by the filename-key instruction. It is the default value in the "Filename" field in the SA Client.

Combining tags on one line

You can combine multiple instruction tags into one instruction by separating the instructions by semicolons and only using a single exclamation point at the beginning as follows:

```
@!namespace=/example/namespace/;filename-key=/files/example;
filename-default=/etc/example@
```

This can be handy, since this one line included in any file makes it a valid CML template, assuming of course, that all other CML tags are correctly formed.

Use case 1 - Simple Key=Value configuration file

The simplest type of configuration file has one or more Key = Value entries as in the following example:

```
Port = 1280
IPAddress = 192.168.0.1
ServerName = server01
```

In this configuration file, types and descriptions are easy to figure out.

Using the Replace instruction

To write a template for this configuration file, use a CML tag to represent where the value exists and where it will be stored in the value set name space. The tag for this is the Replace tag.

The replace tag is composed of several fields and, as with all tags in the CML language, it begins and ends with the “@” symbol and the fields are separated by semicolons. The form looks something like this:

```
@ <name> ; [type] ; [range] ; [option] ; [option] … @
```

Of all the fields, only the `<name>` field is required. So the most basic CML that could represent the configuration file above would be:

```
@!namespace=/example/namespace/@
@!filename-key=/files/example@
@!filename-default=/etc/example@
Port = @port@
IPAddress = @ipaddress@
ServerName = @servername@
```

This specifies that the value for the port number will be stored in the SA database at the key “/example/namespace/port”. The IP address will be stored at the key “/example/namespace/ipaddress” and the server name at the key “/example/namespace/servername”.

This would technically work, but you would be missing a lot of the field validation and error checking available that prevents entering invalid data such as “someport” for the port, for example.

The <name> field in the Replace instruction tag

If the *<name>* field is relative (that is, it does not start with a “/” or a “.”) it gets appended to the current name space and becomes part of the key used to store the value read from the SA database by this tag.

If the name is absolute (that is, it starts with a “/”) it is the entire key and the value gets stored under this key.

Finally, if the name starts with a dot “.”, it will be appended to the name space of whatever loop it is a part of. With very few exceptions, every tag inside a loop should start with a dot, “.”.

The <type> field in the Replace instruction tag

The *<type>* field lets you assign certain predefined ranges and error checking to different values, based on well-known types. For the full list of types, see "CML type attributes" on page 391. For this configuration file, use the predefined types “port,” “ip” and “hostname” for the separate entries, as follows:

```
@!namespace=/example/namespace/@
@!filename-key=/files/example@
@!filename-default=/etc/example@
Port = @port;port@
IPAddress = @ipaddress;ip@
ServerName = @servername;hostname@
```

Adding these types restricts the values and provides validation and error checking.

You can also use the replace tag to represent a sequence of repeating values by prepending “ordered-” or “unordered-” and appending “-set” or “-list”. More on this in the next example.

The default for this field is “string”, which will match anything.

The <range> field in the Replace instruction tag

The *<range>* field allows you to set the allowable range for the values. You can set either integer ranges or string ranges. Integer ranges are valid for any type that consists of strictly integers, string ranges are valid for all other types.

Ranges can be combined with logical OR by using a comma, “,”. Ranges can be combined with logical AND by using the ampersand character, “&”. The “!” character negates the range.

Keep in mind that the specified ranges will be used when reading in a configuration file as well as when accepting values from the SA Client. If you have a configuration file that has a value outside of the ranges you set in the template, then an error will be given when parsing that file. Specify the valid ranges based on the documentation for the configuration file.

Integer ranges

Integer ranges can only use the < and the = symbols to specify "less than" or "greater than" ranges. Specify the position of the number used in the comparison as follows:

Specifying integer ranges

Range condition	Symbols to use
Greater than	n<
Greater than or equal	n<=
Less than	<n
Less than or equal	<=n
Equal	=n

For example, if the ports in the configuration file can only be between 1024 and 2048 inclusive, you would add the ranges to the tag like this:

```
Port = @port;port;1024<=&<=2048@
```

String ranges

String ranges can be a list of valid strings surrounded by quotes and a list of regular expressions starting with the characters r" and ending with a quote. For example, if the ServerName field can only be anything starting with the word "server", you would want to add the ranges to the servername tag like this:

```
ServerName = @servername;hostname;r"server.*”@
```

The [option] fields in the Replace instruction tag

You may append as many options as you need to the tag. Everything after the third semicolon is considered an option and every option is separated by semicolons.

For example, if the IPAddress line in the configuration file is optional and not required to make the configuration file valid, and the IP address could stop at a forward-slash, you could add the option like this:

```
IPAddress = @ipaddress;ip;;optional;delimiter="/"@
```

This would match the following entry:

```
IPAddress = 192.168.0.1
```

It would also match the following entry:

```
IPAddress = 192.168.0.2/
```

Notice that the range field is left empty. Any fields you want to leave as default must still be represented if you want to fill in any later fields. For instance, the following two lines are valid:

```
@ipaddress@

@ipaddress;;;optional@
```

However, the following is not valid because the field “optional” will be interpreted as the <type> field rather than as an option and will result in an error.

```
@ipaddress;optional@
```

The Final CML Template

After all the types, options and ranges are set, the CML template should look something like this:

```
@!namespace=/example/namespace/@
@!filename-key=/files/example@
@!filename-default=/etc/example@
Port = @port;port;1024<=&<=2048@
IPAddress = @ipaddress;ip;;optional;delimiter="/"@
ServerName = @servername;hostname;r"server.*"@
```

Resulting value set

Using the above configuration template to read in the example target configuration file from above will result in the following value set stored in the SA database:

```
/example/namespace/port = 1280
/example/namespace/ipaddress = 192.168.0.1
/example/namespace/servername = server01
```

As you can see, the keys in the name space are a combination of the template's name space (/example/namespace) and each individual tag's name, since they all use relative names.

Use case 2 - Repeating values in the configuration file

It is possible you will encounter a configuration file that will be nothing but a list of values; for example, a file that contains only a list of user names that have write access to a directory. The format of this file could look something like this:

```
admin;
user1;
user2;
```

The repeating lines in this file call for more than the replace tag described in the previous example. The most basic CML that could match this configuration file is the following loop instruction:

```
@!namespace=/wuserlist/namespace/@
@!filename-key=/wuserlistfile/example@
@!filename-default=/etc/wusers.txt@
@*users@
@.@
```

Using the Loop instruction tag

The Loop instruction is used when a set of values may appear multiple times in a configuration file. The default behavior of the loop instruction is to loop over the line of CML directly after it, though this can be modified to loop over multiple lines or within a single line.

The form looks something like this:

```
@ [group level] * <name> ; [ "ordered" | "unordered" ] - [type] - ["set" | "list" ]
; [range] ; [option] ; [option] … @
```

You'll notice some differences and similarities between this tag and the Replace tag in the previous section. The following sections describe each of the options on the Loop tag.

<group level> field

For this example configuration file, group level is not important and it will be discussed later. For now just know that it is used to determine what specific section to loop over, whether within a single line or over multiple lines.

For this example, leave the group level blank which indicates that this loop tag will only iterate over the CML line directly below it.

Instruction type specifier Field, *

The “*” is an instruction type specifier. It signifies what type of instruction this is, in this case, a Loop instruction. The Replace instruction is the default instruction type since it is so common, therefore it does not have an instruction type specifier.

<name> field

The same `<name>` field rules apply to this tag as to the replace tag. To review, see "The <name> field in the Replace instruction tag" on page 365. Any named tag that is part of this loop will need a “.” (dot) in front of the name, and that tag’s namespace gets appended to this loop’s namespace. Likewise, if this loop were a part of another loop, it would need a “.” in front of the name.

This example will use the name “users” as follows:

```
@*users@
```

[type] field

The [`type`] field for the Loop tag is different from the Replace tag’s [*type*] field in two major ways. (To review, see "The <type> field in the Replace instruction tag" on page 365.)

- Ordered vs. Unordered and Set vs. List

  The basic types include all the same types as the Replace tag, but since this will be a repeating sequence of values, information about the sequence needs to be specified. This information is included in this modified [type] field by prepending “ordered” or “unordered”, followed by a dash, and appending a dash followed by “set” or “list” to the type.

  - Prepending “ordered” specifies that the order of the values will be preserved.
  - Prepending “unordered” specifies that the values can be in any order.
  - Appending “set” specifies that the values must be unique.
  - Appending “list” specifies that the values can be repeating.

    While this is optional for the Replace tag, the ordered or unordered option and set or list option is required for the Loop tag.

    For this example, the data is unordered so “set” should be appended, as there would be no reason to order an access list or to have repeating values.

- Namespace Type
  This type is unique to the Loop tag. If the section you are going to be iterating over contains more than one value, then you need to use the name space type.

The default type for this tag is “unordered-string-list”.

For this example the default value would work, but it would be better to specify the type to be “user”. Assuming an unordered set, the resulting tag would look like this:

```
@*users;unordered-user-set@
```

[range] field

Ranges are the same as in the Replace tag for every type except the name space type. Since the name space type iterates over several different tags that may have their own ranges, no range should be used.

Since this example uses the “user” type, it can also use a range. For example, if the documentation for this configuration file were to say that “root” is not a valid user, you could set the range to be valid for anything except root as follows:

```
@*users;unordered-user-set;!"root;"@
```

[option] field

The Option field is the same as the Replace tag's option field. To review, see "The [option] fields in the Replace instruction tag" on page 366.

For this example, we can eliminate the “;” from the user names by setting semicolon as the field delimiter and including it in the line we iterating over to eliminate it from the value that is read in, as follows:

```
@*users;unordered-user-set;!"root";field-delimiter-is-semicolon@
@.@;
```

Loop target tag

The loop target tag looks like the following:

```
@.@
```

Its only purpose is to signify the position of the loop value, which is where the data will be placed in the resulting configuration file.

Final CML

After all the types, options and ranges are set, the CML template should look something like this:

```
@!namespace=/wuserlist/namespace/@
@!filename-key=/wuserlistfile/example@
@!filename-default=/etc/wusers.txt@
```

```
@*users;unordered-user-set;!"root";field-delimiter-is-semicolon@
@.@;
```

Resulting value set

Every value that gets read in will need to be stored in the value set under a unique key. To handle sequences, CML will append a unique number on to the name space starting at 1 and incrementing for each additional iteration.

The resulting value set for the example configuration file using the CML above will look like the following:

```
/example/namespace/users/1 = admin
/example/namespace/users/2 = user1
/example/namespace/users/3 = user2
```

Use case 3 - Complex repeating values in the configuration file

This example models the /etc/hosts file, which is a list of IP addresses followed by a list of host names like this:

```
127.0.0.1 localhost
192.168.0.1 server1 server1.domain.com
192.168.0.2 server2 server2.domain.com
```

This example is similar to the previous example, except that this example iterates over a more complex line. The best way to think about this is by first modeling the single instance of the line using CML Replace instructions like this:

```
@ip-addr;ip@ @sname;unordered-hostname-set@
```

This line defines two replace instructions, one for the IP address and one for the server name.

Notice that the “ip-addr” replace tag specifies the data type as “ip” because it must be an IP address.

The “sname” replace tag is typed as an “unordered-hostname-set”. This means that it can match a list of host names and it will store them along with the corresponding IP address. This is similar to how the Loop tag works and the values get stored in the same way.

This CML is for one iteration. The next step encloses it in a loop. To do this, use the name space loop, since it is iterating over more than one value on each line, and prepend a "." to the names of the tags in the loop, as follows:

```
@*entries;unordered-namespace-set@
@.ip-addr;ip@ @.sname;unordered-hostname-set@
```

The Loop tag (indicated by the @* characters) defines a loop. The "unordered-hostname-set" indicates that the data are host names, the host names can be in any order, and the values must be unique.

The "." characters added before the "ip-addr" and "sname" strings in the Replace instruction indicate that these are the target of the Loop instruction.

Final CML

Adding the above loop and replace CML to the required name space and file lines gives the following.

```
@!namespace=/example/namespace/@
@!filename-key=/files/example@
@!filename-default=/etc/hosts@
@*entries;unordered-namespace-set@
@.ip-addr;ip@ @sname;unordered-hostname-set@
```

Resulting value set

Below are the values that will be stored in the SA database if the entries in the sample file are read in using the SA Client.

```
/example/namespace/entries1/ip-addr = 127.0.0.1
/example/namespace/entries1/sname/1 = localhost
/example/namespace/entries2/ip-addr = 192.168.0.1
/example/namespace/entries2/sname/1 = server1
/example/namespace/entries2/sname/2 = server1.domain.com
/example/namespace/entries3/ip-addr = 192.168.0.2
/example/namespace/entries3/sname/1 = server2
/example/namespace/entries3/sname/2 = server2.domain.com
```

Partial templates

While it is best to model the entire configuration file, you can use partial templates to model only part of a configuration file. To create a partial template, you must have a copy of the full configuration file on a server for the template to read. And you must use the Preserve Format option to preserve the rest of the file.

The following shows a simple configuration file.

```
UserName = alice
Password = pass
HomeDir = /home/alice
```

To manage only the home directory line, use the `@!partial-template` instruction and model only the line you want to manage. The template would look like this:

```
@!namespace=/example/@
@!filename-key=/files/example@
@!filename-default=/usr/example@
@!partial-template@
HomeDir = @homedir;dir@
```

For more information on the Preserve Format setting, see "Setting values in the Value Set Editor" on page 232. See also "The @!full-template and @!partial-template attributes" on page 401.

CML Reference

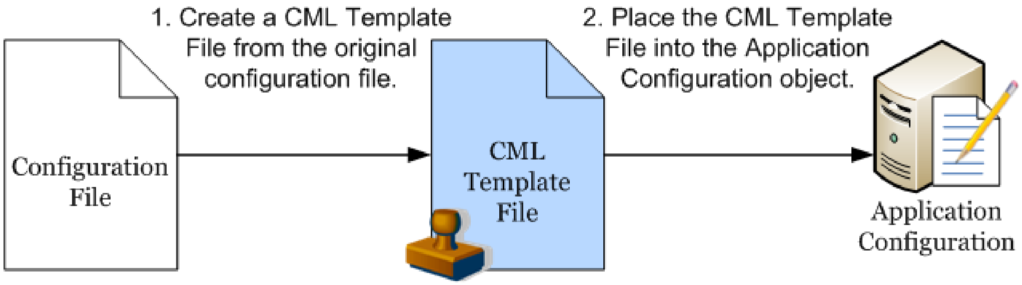
The Configuration Modeling Language (CML) is used to create a **template** of a **configuration file** so it can be managed from SA. A CML template is a separate file you create that models the format of the configuration file so the variable values in the configuration file can be set to different values for different sets of servers.

The template file contains data, directives and definitions so that an actual configuration file can be generated from the template and a set of values.

CML defines a two-way transform: it specifies how to move values from a configuration file to a value set in the SA database, and it specifies how data from a value set is merged with the template to create a properly formatted configuration file that can be pushed to a managed server.

You also write scripts using CML that are run when pushing configurations to managed servers. For more information, see "Running scripts with Application Configurations" on page 244

CML Template


1. Create a CML Template File from the original configuration file.
2. Place the CML Template File into the Application Configuration object.
Configuration File
CML Template File
Application Configuration

XML Configuration Files

SA can also manage XML configuration files. For more information on using XML configuration templates, see "Managing XML configuration files" on page 298.

Configuration templates

A configuration template is a "templatized" version of an actual configuration file whose values have been turned into variables. Using the SA Client, you can define a template's value sets, save them to the SA database, and then propagate those values to a real configuration file on a managed server.

Value sets are stored on the SA database. Storing all values in the SA database allows you to manage configuration values from a central location and ensures configuration consistency across applications in your data center.

Once the template version of the configuration file has been created and added to an application configuration object and you have created a value set for the template, you can push those values to configuration files on managed servers.

CML overview

A configuration template consists of a series of CML tags. Each tag represents either an instruction to the CML parser how to interpret the text in the configuration file or a placeholder that identifies the location of a value in the configuration file and how to map it into a value set.

Keep in mind that the configuration template contains no values. It only defines how values are moved between the value set in the SA database and the configuration file instances on the managed servers. For more information, see " Value sets" on page 229.

Template files containing CML are typically named with ".tpl" as the file extension, but this file extension is not required.

Structure of CML tags

The basic building blocks of a CML tag is as follows:

```
@{level}{tagtype}{source};{type};{range};{option};...;{option}@
```

> **Note:**
> Neither whitespace nor '@' can appear inside a CML tag. The '@' symbol can be escaped by prepending it with another '@'.

The following rules apply to all CML tag:

- All CML tags start and ends with the @ symbol.
- Semicolons (;) mark placeholders for omitted attributes. For example, the following shows two omitted attributes between the @name attribute and the optional@ attribute:

  ```
  @name;;;optional@
  ```
- If attributes to the right of a semicolon are empty, then semicolons are optional. For example: `@name@`
- **{level}** - The block level is an integer that specifies the nesting level of the block. The level also determines whether the block spans multiple lines or is part of a single line. If the level is between 1 and 99 it is a multi-line block. If it is above 101, it is a block within a line. Each block open tag closes all previous blocks that have an equal or greater level.

  Do not use level 100 because level 100 is reserved.
- {**tagtype**} - CML defines the tag types listed below. Each type is an instruction to the CML parser. For complete details, see "CML tag types" on page 378.
  - Comment Tags: @# and @## ... #@ - Define comments in the template.
  - Replace Tag: @ - Defines how to replace a variable with a value from the value set.
  - Instruction Tag: @! - Gives an instruction to the CML parser.
  - Block Tags: @[@...@]@ - Create a new scope.

  - Loop Tag: @* - Creates a loop over multiple similar values.
  - Loop Target Tag: @. - Ends a loop.
  - Conditional Tag: @?
  - DTD Tag: @~ - Defines
- {**source**} - Defines a key where the value is stored in the value set. Absolute path names start with a “/” character. Relative path names do not start with a “/” and are concatenated to the name space key value defined by the `@!namespace` instruction.
- {**type**} - Defines the data type of the value required by the configuration file and the corresponding value in the value set. For example, int for integer, string, boolean, IP-address, and so forth. You can also specify ordered and unordered lists and sets.
- {**range**} - Defines additional restrictions on data values for better error checking.
- {**option**} - Defines additional parameters you can specify to modify the behavior of the CML tag.

Required CML tags

Every CML file must define its name space and the default file name of the configuration file the template models using the following CML tags:

- `@!namespace` defines the name space for the template. All values in the value set used by the template will be stored in the SA database at the key defined by the `@!namespace` tag. For more information, see "Define the Namespace with the @!namespace CML Tag" below.
- `@!filename-key` defines a specific key where the default file name will be stored in the SA database. This key can either be a separate name space or it can be appended to the name space defined by the `@!namespace` tag. For more information, see "Defining the default configuration file name with the @!filename-key and @!filename-default CML tags" on the next page.
- `@!filename-default` defines the directory and name of the configuration file being modeled by the template. This value can be modified by the value set. For more information, see "Defining the default configuration file name with the @!filename-key and @!filename-default CML tags" on the next page.

Define the Namespace with the @!namespace CML Tag

The name space in a CML template file defines a unique key value where data is stored in the database. The name space value is represented as a path name and looks like a directory path name in a file system, or a URI in a web browser’s location bar. Use the `namespace` tag to define the name space.

The path names for individual values can be either absolute or relative. An absolute path name start with “/” and is the complete representation of the location of the value in the value set. A path name that does not start with a “/” is a relative path name; its value will be appended to the current value of the name space.

The `namespace` tag is required.

All key names in CML templates must be ASCII. Other fields and text can be either ASCII or non-ASCII text.

Below is an example of a `namespace` tag in a CML template:

```
@!namespace=/security/@
```

Defining the default configuration file name with the @!filename-key and @!filename-default CML tags

Each template must define the default configuration file name that will be used when pushing the generated configuration file to a server. This file name can be overridden by the value sets. Use the `filename-default` tag to define the default file name.

You must also specify a unique key where the default file name will be stored in the SA database. This key can be combined with the name space to generate a unique storage location for the default file name. The key defines a name space is represented as a path name. Use the `filename-key` tag to define the key value.

The `filename-default` and `filename-key` tags are required.

All key names in CML templates must be ASCII. Other fields and text can be either ASCII or non-ASCII text.

The following example CML specifies that they key value “/files/hosts” will be used to store the default file name in the SA database. It also specifies that the default file name for the generated configuration file will be “/etc/hosts”.

```
@!filename-key="/files/hosts"@
@!filename-default="/etc/hosts"@
```

You can also combine CML tags on one line as follows:

```
@!filename-key="/files/hosts";filename-default="/etc/hosts"@
```

Example CML template for /etc/hosts

The following is an example of a CML template that models a typical /etc/hosts file.

```
@#############################################
#                                            #
# /etc/hosts (multiplatform)                 #
# Version 2.0                                #
# Joe Author (joe_author@your_company.com)   #
#                                            #
#############################################@
@!namespace=/system/dns/@
@!filename-key="/files/hosts";filename-default="/etc/hosts"@
@!unordered-lines;missing-values-are-error@
@!relaxed-whitespace@
@!sequence-delimiter-is-whitespace@
@!line-comment="#"@
@~host/.ip
type = ip
printable = IP address
description = This is an IP address
@
@~host/.hostnames
type = unordered-hostname-set
printable = Hostnames
description = A set of hostnames
@
@1*host;unordered-namespace-set;;sequence-append@
@.ip@   .hostnames@
@1]@
```

CML tag types

The following are the main CML tags. These are described in detail below.

- "Comment Tag: @# and @##" on the next page
- "Replace Tag: @" on page 380
- "Instruction Tags: @!" on page 381
- "Block (or Group) Tag: @[@...@]@" on page 382
- "Loop Tag: @*" on page 385
- "Loop Target Tag: @." on page 387
- "Conditional Tag: @?" on page 387
- "DTD Tag: @~" on page 389

Comment Tag: @# and @##

This tag defines a comment in the CML template file. You can define one line comments or multiple line comments.

Syntax

```
@#   <one line comment>
```

Or:

```
@## <comments spanning multiple lines>
    <comments spanning multiple lines>
    <comments spanning multiple lines> #@
```

Description

The comment tag can be used to insert comments anywhere in your CML file.

As a best practice, use the comment tag at the beginning of a CML template to create a header describing the template, such as the name of the template, the configuration file the template is based on, the purpose of the template, the author, the date, and so on.

Attributes

None.

Examples

The following is a one line comment:

```
@# This comment ends at the end of this line.
```

The following is a multiple line comment:

```
@##

 This comment spans

   multiple lines.

#@
```

The following is also a multiple line comment:

```
@#########################################################
# /etc/hosts (multiplatform) #
# $Id: hosts.tpl 8650 2006-06-05 05:28:03Z joe_author $
##########################################################@
```

Replace Tag: @

This tag replaces text in the template file with a value from the value set.

Syntax

```
@{source}[;[{type}][;[{range}[;{option}[;{option}]...]]]]@
```

Description

The replace tag replaces the tag in a CML line with the data from the specified location in the name space. It is an indicator that the text in this location is data, and it also specifies details about how that data should be stored and validated. The source name is the index key where the data is found in the value set. The other fields of the replacement tag specify details about how the data should be stored and validated.

The replace tag is the only tag that is not indicated by a special character following the "@" character. The only required element in a replace tag is the source. All other elements are optional.

Attributes

- **Source**: The source attribute is the key used to store and access the value in the value set. If the source attribute is relative (that is, it does not start with a "/" or a ".") it gets appended to the current name space and becomes part of the key used to store the value read by this tag. If the name is absolute (that is, it starts with a "/") it is the key, and the value gets stored under this key.
- The only required element in a replace tag is the source. All other elements are optional. If the name starts with a ".", it will be appended to the name space of the loop it is a part of. Tags inside a loop typically start with a ".".
- **Type**: The type attribute specifies the type of the replace tag, which applies certain predefined restrictions and error checking to different values. The default type for replace tags is "string".
- The available types are described at "CML type attributes" on page 391.
- **Range**: The range attribute allows you to set the range of valued values. (Keep in mind that all ranges will be used when reading in a file as well as when accepting values from a user.) If you have a configuration file that has a value outside of the specified ranges, then an error will occur when parsing that file.
- Ranges are described at "CML range attributes" on page 396.

- **Options**: The option attributes modify the behavior of the tag. Multiple options can be appended to the end of most tags, separated by semicolons. Everything after the third semicolon is considered an option. Options can also be used as instruction tags.

Options are described at "CML global option attributes" on page 400 and "CML regular option attributes" on page 402.

Example 1

```
Title=@main_title@
```

In this example, `main_title` will extract the string that follows “`Title=`“ text in the configuration file, and store it at key location /main_title in the value set.

Or if you are performing a push, `main_title` will extract the value stored from location `/main_title` from the value set, and push it after the string “`Title=`“ text in the configuration file.

Example 2

```
Port = @port;port;1024<=&<=2048@
IPAddress = @ipaddress;ip;;optional;delimiter=”/”@
ServerName = @servername;hostname;”localhost”,r”server.*”@
```

Instruction Tags: @!

This tag specifies parser actions. For a list of available instructions, see "CML global option attributes" on page 400 and "CML regular option attributes" on page 402.

Syntax

```
@!{option}[[;{option}]...]@
```

Description

The instruction tag sets options that will be used at parse time. For example, defining the name space, whether a list is sorted, ordered, or unordered, how the parser should interpret whitespace, acceptable delimiters, defining comment characters, and so on.

The only attributes used by an instruction tag are options. One or more option can appear in one instruction tag. Multiple options are separated by semicolons. To understand how any particular instruction tag affects the parser, refer to the descriptions of the embedded options.

Attribute

Only option attributes are used with an instruction tag.

Options: The option attributes in an instruction tag define the behavior of the tag. Multiple options can be appended to the end of most tags, separated by semicolons. Many options are toggles of other options. When an option from one of these toggling groups appears in a block, no other option from that group should appear in the same block.

Options are described at "CML global option attributes" on page 400and "CML regular option attributes" on page 402.

Example 1

The following instruction tag tells the CML parser that whitespace in the template will be matched by any combination of tabs and spaces.

```
@!relaxed-whitespace@
```

Example 2

The two options in the following instruction tag tell the CML parser the relative order of lines in the configuration file is not important to mapping values from those lines with the value set and it is not an error if values in the value set are not matched by text in the configuration file.

```
@!unordered-lines;missing-values-are-null@
```

Example 3

```
@!namespace=/test/@
@!filename-key="/test";filename-default="/tmp/test.txt"@
@!optional-whitespace@
@!boolean-yes-format="1";boolean-no-format="0"@
@!line-comment-is-semicolon@
@!unordered-lines@
```

Block (or Group) Tag: @[@...@]@

The Block tag is sometimes also referred to as the Group tag. This tag creates a block or group of related tags and lets you nest groups of related tags.

Syntax

The block tag can have either single line syntax or multiple line syntax.

Single line syntax for the block tag is as follows (with literal strings quoted):

```
"@" [{level}] "[" [ ";" {option}[ ";" {option}]...] "@" {CML statements} "@"
[{level}] "]@"
```

Multiple line syntax for the group tag is as follows:

```
"@" {level} "[" [ ";" {option}[ ";" {option}]...] "@"
{CML statements}
"@" {level} "]@"
```

@[{level}][[;{option}[;{option}]...]@ {set of CML tags} @[{level}]]@

@[{level}][[;{option}[;{option}]...]@

{set of CML tags}

@[{level}]]@

The level is an integer that determines whether the block spans multiple lines or is part of a single line. If the level is between 1 and 99 it is a multiline block. If it is greater than 101, it is a block within a line. Do not use level 100 because it is reserved.

Each block can be ended explicitly or implicitly. To end a block explicitly, use an end block tag with the level number. For example, the following tag explicitly closes a level 3 block: `@3]@`.

To end a block implicitly, use an end block tag with a lower level number to end an enclosing block, or define a new block with a lower level. Each block open tag will close all previous blocks that have an equal or greater level.

Description

The block tag allows you to group related tags and nest groups of tags. With a block, you can define separate parsing rules for each section of a configuration file.

You can nest blocks within other blocks using a higher number for the level. Any subsequent tag with a level value will close all open levels of equal or great value. The block close tag, `@]@`, is not required.

The opening block tag can include option attributes. Those attributes only affect the tags inside the block at the level declared by the opening tag. Contrast that with instruction tags that appear inside the block: those instruction tags affect the behavior of the current level and any nested blocks.

By using blocks, you can specify unique options for each separate section of the configuration file. For example, you might have a section of a configuration file where the values for True and False are defined as “1” and “0”, respectively. In another section in the same file, you could define values for True and False as “T” and “F”. Use the block tag to separate the two different ways of defining True and False.

Another example could be if in one section of a configuration file a specific number of spaces are important, while in another section any number of spaces is acceptable. You can use the block tag to indicate where the number of spaces differ.

Attributes

No name, type, or range attributes are used with block tags.

- **Level**: The block level is an integer that specifies the nesting level of the block. The level also determines whether the block spans multiple lines or is part of a single line. If the level is between 1 and 99 it is a multi-line block. If it is above 101, it is a block within a line. Each block open tag closes all previous blocks that have an equal or greater level.

  > **Note:**
  > Do not use level 100 because level 100 is reserved.

- **Options**: The option attributes modify the behavior of the CML tags in the block. Instruction tags within the block affect the behavior of CML tags in the current block and in nested blocks. Multiple options can be appended to the end of most tags, separated by semicolons. Options can also be used as instruction tags.

  Options are described at "CML global option attributes" on page 400 and "CML regular option attributes" on page 402.

Example 1

The following example creates two blocks, one block nested within the other. The first line defines the first block, which is the outer block. The fourth line defines the second block, which is the inner block nested within the first block. The second to last line closes the inner block. This line is optional. The last line closes the outer block. If the second to last line were omitted, the last line would close both blocks.

```
@1[@
@!ordered-lines@
[SectionOne]
@2[@
@!unordered-lines@
optionA = @section_one/option_a@
optionB = @section_one/option_b@
@2]@
@1]@
```

Example 2

This example models two sections named [Options] and [AllowVerbs] in a Windows `UrlScan.ini` file. Both sections in this file contain a list of key-value pairs.

To define the first section (lines 1 through 3), you can use the block tag (`[`) set at two levels because there are two kinds of data in this section: a fixed heading followed by a list of key-value pairs. The first level block handles the text string “[Options]” while the second level block handles all of the key-value pairs in that section.

The second section (lines 4 through 6) defines the [AllowVerbs] section. Notice that the first section is not explicitly closed with the `@2]@` and `@1]@` tags as in the previous example because opening the next level 1 section (line 4) implicitly closes the previous sections.

```
@1[;optional;ordered-lines@
[Options]
@2[;unordered-lines@
@1[;optional;ordered-lines@
[AllowVerbs]
@2[;unordered-lines@
```

Loop Tag: @*

This tag defines a processing loop. See also the "Loop Target Tag: @." on page 387.

Syntax

```
@[{level}]*{source}[;[{type}][;[{range}[;{option}[;{option}]...]]]] @ {target}
```

Description

The Loop tag is used when a set of values may appear multiple times in a configuration file. The default behavior of the loop tag is to iterate over the line of CML directly after it, though this can be modified to iterate over multiple lines or within a single line.

Loops are a form of Group tag, see the Group tag for more information.

The Loop tag allows sequences (lists and sets) to be enumerated. The block associated with a loop element will be processed for each incident of that block in an input file, and will be generated in an output file for each incidence of that data in a value set.

The group associated with a loop element will cause a new element to be stored in the value set for each incident of that group in a configuration file, or each incidence of that data in a value set will push a value to the configuration file. The source attribute is the index key used to map values in the value set.

Attributes

- **Level**: The group level is an integer that determines whether the group spans multiple lines or is part of a single line. If the level is between 1 and 99 it is a multi-line group. If it is above 101, it is a group within a line. Level 100 is reserved for internal purposes. Each group open tag will close all previous groups that have an equal or greater level.

- **Source**: The source attribute is the key used to access the value in the value set. If the source attribute is relative (that is, it does not start with a “/” or a “.”) it gets appended to the current name space and becomes part of the key used to store the value read in by this tag. If the name is absolute (that is, it starts with a “/”) it is the key, and the value gets stored under this key. The only required element in a loop tag is the source; everything else is optional. If the source name starts with a “.”, it is to be appended to the name space of whatever loop it is a part of. Tags inside loops typically start with a “.”.
- **Type**: The type attribute specifies the type of the replace tag, which applies certain predefined restrictions and error checking to different values. The default type for replace tags is “string”.

  For the full list of types see "CML type attributes" on page 391.

  You can prepend “ordered-” or “unordered-” to the type. And you can append “-set” or “-list” to the type.
  - Prepending “ordered-” specifies that the values must be in order.
  - Prepending “unordered-” specifies that the values can be in any order.
  - Appending “-set” specifies that the values must be unique.
  - Appending “-list” specifies that the values can be repeating.
- **Range**: The range attribute allows you to set the range for the values. Keep in mind that all ranges will be used when reading in a file as well as when accepting values from a user. If you have a configuration file that has a value outside of the specified ranges, then an error will occur when parsing that file.
- Ranges are described at "CML range attributes" on page 396.
- **Options**: The options attributes modify the behavior of the tag. Multiple options can be appended to the end most tags, separated by semicolons. Everything after the third semicolon is considered an option. Options can also be used as instruction tag.

  Options are described at "CML global option attributes" on page 400 and "CML regular option attributes" on page 402.

Example 1

The asterisk character indicates a loop tag. For example:

```
@1*includegroup;ordered-namespace-set;;optional@
#BEGIN_ALTERNATE
@*.include@
#INCLUDE @.@
```

```
#END_ALTERNATE
@1]@
```

Example 2

```
@*users;unordered-user-set;!”root”;field-delimiter-is-semicolon@
@.@;
```

Loop Target Tag: @.

The loop target tag defines an iteration for the Loop tag. See "Loop Tag: @*" on page 385.

Syntax

```
@.[{source}[;[{type}][;[{range}[;{option}[;{option}]...]]]]]@
```

Description

The loop target tag indicates the placeholder for a value in a loop. If you consider that the loop tag indicates the beginning of a loop, and is therefore similar to a group tag, the loop target tag is quite similar to a replace tag.

When encountered in a group, with each loop iteration, this tag simply maps the text at current position in the configuration file with the current value in value set. If the optional source attribute is used, the source is appended to the name space created by the loop.

Attributes

None.

Example

The loop target tag is indicated by a period following the “@” character. For example:

```
@*keys;unordered-namespace-set@
@.key@ = @.value@
```

Conditional Tag: @?

This tag defines a condition.

Syntax

```
@[{level}]?{source}@{text}
```

Description

The conditional tag maps whether or not the text exists in the configuration file with a Boolean value in the name space. When reading a target configuration file, if the text matches, the name space value gets true, otherwise false. When writing to a configuration file, if the name space value is true then the configuration files gets the text; otherwise, no text is written.

This is one of the few tags in which something outside of the tag is actually the value. The main use of this tag is to store a Boolean true value in a location in the name space if the text after the tag exists.

Attributes

No type, range or option attributes are used with conditional tags.

- **Level**: The level is an integer that determines whether the group spans multiple lines or is part of a single line. If the level is between 1 and 99 it is a multi-line group, if it is above 101, it is a group within a line. Level 100 is reserved for internal purposes. Each group open tag will close all previous groups that have an equal or greater level.
- **Source**: The source attribute is the key used to access the Boolean value. If the source attribute is relative (does not start with a "/" or a ".") it gets appended to the current name space and becomes part of the key used to store the value read in by this tag. If the name is absolute (starts with a "/") it *is* the key, and the value gets stored under this key.

  If the source name starts with a ".", it is to be appended to the name space of whatever loop it is a part of. Typically a tag inside a loop should start with a ".".

Example 1

The conditional tag is indicated by a question mark symbol (?). For example:

```
@?debug@options debug
```

In this example, if you were importing a configuration file into a configuration template, and if the text "options debug" exists in the configuration file, then the value at key `/debug` will be set to true.

If you were going to push the application configuration, if the value stored at key **/debug** is true, then the text "options debug" will be pushed to the configuration file.

Example 2

For example, if a configuration file specified that an application were to be threaded based on the existence of the key word "threaded" in the configuration file, the CML would look like this:

```
@?is_threaded@threaded
```

This sets the value at the name space key `/is_threaded` to true if the value “threaded” is in the configuration file and to false if the value “threaded” is not in the configuration file.

DTD Tag: @~

This tag defines a DTD.

Syntax

```
@~{source}
[type = {type}]
[description = {description}]
[printable = {printable}]
[range = {range}]
[{option}
...]
@
```

Description

CML supports Document Type Definition (DTD) tags that can be used to pre-define attributes for other CML tags. DTD’s can be used to make the actual functional part of the CML template a little cleaner by storing all of the characteristics of the tag in another location and just referencing the tag itself by name.

DTD definitions can be used to define any tag that has a source attribute; for example loop tags, loop target tags, replace tags, but not tags like instruction tags or group tags (which do not have a source attribute).

Another advantage of using DTD tags in CML is the ability to define 'PRINTABLE' and 'DESCRIPTION' values. The 'PRINTABLE' and 'DESCRIPTION' values give the user some feedback regarding the intended purpose of the field. The string value of the DESCRIPTION attribute is displayed when the mouse cursor rolls over the field in the value set editor screen. The string value of the Printable attribute will replace the path name in the value set editor with a easier-to-read field label.

DTD tags in CML are also inherently multi-line tags. All but the first and last line can be in any order, and all the elements here relate to the fields in a tag, except for printable and description, as those two are valid only for DTD defined tags.

For more information on using DTD tags in your configuration templates, see "Use DTD tags in CML" on page 415. For XML templates, see "Customize XML DTD element display" on page 303.

Attributes

No level attribute is used with a DTD tag. The only required attribute in a DTD tag is the source; everything else is optional. However, a DTD tag with only the name defined does nothing useful.

- **Source**: The source attribute is the key used to access the value. If the source attribute is relative (does not start with a “/” or a “.”) it gets appended to the current name space and becomes part of the key used to store the value read in by this tag. If the name is absolute (starts with a “/”) it *is* the key, and the value gets stored under this key. If the source name starts with a “.”, it is to be appended to the name space of whatever loop it is a part of. Typically a tag inside a loop should start with a “.”.
- **Type**: The type attribute assigns certain predefined restrictions and error checking to different values, based on well-known types. The default type for replace tags is “string”, which will match more or less anything.
- The full list of types is available at "CML type attributes" on the next page of this document.
- **DESCRIPTION**: The value of the description attribute is a string that is a brief description of what kind of value this tag represents. This attribute will be displayed as mouse-over text in the SA Client value set editor.
- **PRINTABLE**: The value of the printable attribute is a string that is just a clean name for the variable. It will be displayed in the SA Client value set editor as the name for the attribute.
- **Range**: The range attribute allows you to set the range for the values. You need to keep in mind that all ranges will be used when reading in a file as well as when accepting values from a user. If you have a configuration file that has a value outside of the ranges you set in the template, then an exception will probably get thrown when parsing that file. It is best to use ranges that are correct based on the documentation for the configuration file.
- Ranges are described fully at "CML range attributes" on page 396.
- **Options**: The option attributes serve to modify or affect the behavior of the tag. Multiple options can be appended to the end most tags, separated by semicolons. You may append as many options as you need to the tag, everything after the third semicolon is considered an option, and every option is separated by semicolons. Options can also be used as instruction tag.

  Options are full described at "CML global option attributes" on page 400 and "CML regular option attributes" on page 402.

Example

```
@~port
type = port
range = 1024<=&<=2048
printable = Port
description = The port used for this application. It
should be a port number between 1024 and 2048
@
```

CML type attributes

CML attributes define and control the semantics of a CML tag. This section defines the types you can use in a CML template. Note that some types can be modified to represent a sequence of repeating values by appending “-set” or “-list” to the type. Some types can be modified to ignore the order of a sequence of repeating values by prepending “ordered-“ or “unordered-“ to the type.

The int type

Int is a numeric type.

Syntax

```
@[{level}]{tag-type}[[{source}][;int][;[{range}][;{option}[;{option}]...]]]]@
```

Description

An Integer value …, -2, -1, 0, 1, 2, … (Z).

The decimal type

Decimal is a numeric type.

Syntax

```
@[{level}]{tag-type}[[{source}][;decimal][;[{range}][;{option}[;{option}]...]]]]@
```

Description

Decimal number.

The guid type

Guid is a numeric type.

Syntax

```
@[{level}]{tag-type}[[{source}][;guid][;[{range}][;{option}[;{option}]...]]]]@
```

Description

Globally Unique Identifier (GUID), 128-bit id.

The string type

String is a non-numeric type.

Syntax

```
@[{level}]{tag-type}[[{source}][;string][;[{range}][;{option}[;{option}]...]]]]@
```

Description

String is the default type for all values if no other type is explicitly specified.

The quotedstring type

Quotedstring is a non-numeric type.

Syntax

```
@[{level}]{tag-type}[[{source}][;quotedstring][;[{range}][;{option}[;
{option}]...]]]]@
```

Description

Quoted string.

The boolean type

Boolean is a non-numeric type.

Syntax

```
@[{level}]{tag-type}[[{source}][;boolean][;[{range}][;{option}[;{option}]...]]]]@
```

Description

Boolean.

The duration type

Duration is a non-numeric type.

Syntax

```
@[{level}]{tag-type}[[{source}][;duration][;[{range}][;{option}[;{option}]...]]]]@
```

Description

Duration.

The ipv6 type

IPv6 is a system-specific type.

Syntax

```
@[{level}]{tag-type}[[{source}][;ipv6][;[{range}][;{option}[;{option}]...]]]]@
```

Description

CML supports the following two conventional forms for representing IPv6 addresses as text strings:

x:x:x:x:x:x:x:x, in which “x” represent one to four hexadecimal digits of the eight 16-bit pieces of the IPv6 address. For example:

`ABCD:EF01:2345:6789:ABCD:EF01:2345:6789`

“::” in an IPv6 address indicates one or more groups of 16 bits of zeros. The “::” can appear only once in an address. The “::” can also be used to compress leading or trailing zeros in an address. For example:

`2001:DB8:0:0:8:800:200C:417A` becomes `2001:DB8::8:800:200C:417A`

`0:0:0:0:0:0:0:1` becomes `::1`

The ipv4 type

IPv4 is a system-specific type.

Syntax

```
@[{level}]{tag-type}[[{source}][;ipv4][;[{range}][;{option}[;{option}]...]]]]@
```

Description

IP v4 Address.

The ip type

Ip is a system-specific type.

Syntax

```
@[{level}]{tag-type}[[{source}][;ip][;[{range}][;{option}[;{option}]...]]]]@
```

Description

IP Address (ipv4 and ipv6).

The hostname type

Hostname is a system-specific type.

Syntax

```
@[{level}]{tag-type}[[{source}][;hostname][;[{range}][;{option}[;{option}]...]]]]@
```

Description

The name of a host server.

The host type

Host is a system-specific type.

Syntax

```
@[{level}]{tag-type}[[{source}][;host][;[{range}][;{option}[;{option}]...]]]]@
```

Description

Host IP Address or Hostname.

The network type

Network is a system-specific type.

Syntax

```
@[{level}]{tag-type}[[{source}][;network][;[{range}][;{option}[;{option}]...]]]]@
```

Description

IP v4 Network.

The port type

Port is a system-specific type.

Syntax

```
@[{level}]{tag-type}[[{source}][;port][;[{range}][;{option}[;{option}]...]]]]@
```

Description

TCP or UDP Port.

The user type

User is a system-specific type.

Syntax

```
@[{level}]{tag-type}[[{source}][;user][;[{range}][;{option}[;{option}]...]]]]@
```

Description

Username.

The group type

Group is a system-specific type.

Syntax

```
@[{level}]{tag-type}[[{source}][;group][;[{range}][;{option}[;{option}]...]]]]@
```

Description

Group name.

file – system specific type

Syntax

```
@[{level}]{tag-type}[[{source}][;file][;[{range}][;{option}[;{option}]...]]]]@
```

Description

File name.

The dir type

Dir is a system-specific type.

Syntax

```
@[{level}]{tag-type}[[{source}][;dir][;[{range}][;{option}[;{option}]...]]]]@
```

Description

Directory path name.

The email type

Email is a system-specific type.

Syntax

```
@[{level}]{tag-type}[[{source}][;email][;[{range}][;{option}[;{option}]...]]]]@
```

Description

Email address.

CML range attributes

CML attributes define and control the semantics of a CML tag. This section defines the possible range attributes you can use in a CML template. For a given a CML type, range attributes allow you to define and restrict valid values for tag, using range specifiers.

! & , – Logical operators

! – not specifier

& – and specifier

, – or specifier

Syntax

```
@[{level}]{tag-type}[[{source}][;[{type}][;!{range}][;{option}[;{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;{range}&{range}][;{option}[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;{range},{range}][;{option}[;
{option}]...]]]]@
```

Description

Range specifiers can be modified by logical operators to control how input is validated. The three available operators (in order of precedence) are: not, and, or.

- The not operator is represented with an exclamation point, and is a prefix unary operator. It negates the meaning of the range, meaning that items that satisfy the range return false, and items that fail to satisfy the range return true.
- The and operator is represented with an ampersand, and is an infix binary operator. It returns true if and only if both operands return true.
- The or operator is represented with a comma, and is an infix binary operator. It returns true if and only if either operand returns true.

Whitespace is not significant when specifying ranges.

> **Note:**
> The current CML parser requires that neither whitespace nor ‘@’ appear inside a CML tag.

n< n<= <n <=n =n – Comparison specifiers

n< – greater than specifier

n<= – greater than or equal specifier

<n – less than specifier

<=n – less than or equal specifier

=n – equal specifier

Syntax

@[{level}]{tag-type}[[{source}][;[{type}][;{number}<][;{option}[;{option}]...]]]]@

@[{level}]{tag-type}[[{source}][;[{type}][;{number}<=][;{option}[;{option}]...]]]]@

@[{level}]{tag-type}[[{source}][;[{type}][;<{number}][;{option}[;{option}]...]]]]@

@[{level}]{tag-type}[[{source}][;[{type}][;<={number}][;{option}[;{option}]...]]]]@

@[{level}]{tag-type}[[{source}][;[{type}][;={number}][;{option}[;{option}]...]]]]@

Description

The available specifiers for numeric values are: greater than, greater than or equal to, less than, less than or equal to, and equals.

A greater than specifier (n<) consists of a number, followed by an open angle bracket character. This range is satisfied by numeric values that are greater than the specified number.

A greater than or equal to specifier (n<=) consists of a number, followed by an open angle bracket character, followed by an equals character. This range is satisfied by numeric values that are greater than or equal to the specified number. (Note that for a number n, n<= is equivalent to !<n, and also equal to n<,=n, and is provided for convenience)

A less than specifier (<n) consists of an open angle bracket character, followed by a number. This range is satisfied by numeric values that are greater than the specified number.

A less than or equal to specifier (<=n) consists of an open angle bracket character, followed by an equals character followed by a number. This range is satisfied by numeric values that are greater than or equal to the specified number. (Note that for a number n, <=n is equivalent to !n<, and also equal to <n,=n, and is provided for convenience)

An equals specifier (=n) consists of an equals character, followed by a number. This range is satisfied by numeric values that are equal to the specified number.

It is suggested that when providing two range specifiers separated by an and operator, the greater than (or equal to) specifier precede the less than (or equal to) specifier, for example, 0<=&<256.

Whitespace is not significant when specifying ranges.

> **Note:**
> The current CML parser requires that neither whitespace nor ‘@’ appear inside a CML tag.

" – String literal specifier

Syntax

```
@[{level}]{tag-type}[[{source}][;[{type}][;"{string}"][;{option}[;{option}]...]]]]@
```

Description

A string literal specifier consists of a double quote character, followed by a string of text, followed by a double quote character. The quoting and escaping rules follow those of the C language; that is, that embedded quotes are escaped with a backslash, a newline is represented by \n, a tab character is represented by \t, and a literal backslash is represented by \\. This range is satisfied by string values that exactly match the text.

Whitespace is not significant when specifying ranges.

> **Note:**
> The current CML parser requires that neither whitespace nor '@' appear inside a CML tag.

r" – Regular expression specifier

Syntax

```
@[{level}]{tag-type}[[{source}][;[{type}][;r"{regular expression}"][;{option}[;
{option}]...]]]]@
```

Description

A regular expression specifier consists of the "r" character, a double quote character, followed by a regular expression, followed by a double quote character ("). The quoting and escaping rules follow those of Python regular expressions, with the exception of the quote character, which must be escaped with a backslash character. This range is satisfied by string values that match the regular expression.

Whitespace is not significant when specifying ranges.

> **Note:**
> The current CML parser requires that neither whitespace nor '@' appear inside a CML tag.

CML global option attributes

CML attributes define and control the semantics of a CML tag. This section defines the possible global attributes you can use in a CML template. Global options can only be used in instruction tags, and cannot be used as option attributes in other tag types.

The @!filename-key attribute

Syntax

`@!filename-key={key}@`

`{key}` has no default value.

Description

`filename-key` identifies a path to the key in a value set that will contain the file name of the file being generated during a push.

The `filename-key` value is a pathname. It can be written as a relative path and does not need to begin with a slash (/).

The filename-key value must not end with a /. This requirement may be relaxed in later versions.

The @!filename-default attribute

Syntax

`@!filename-default={filename}@`

`{filename}` has no default value.

Description

`filename-default` identifies the default filename that will be returned if there is no filename in the Value Set. For example, the user may enter a filename in the Value-Set Editor, thus overriding the filename-default value.

The @!full-template and @!partial-template attributes

Syntax

`@!full-template@`

`@!partial-template@`

`full-template` is the default behavior.

Description

`full-template` is the default behavior and indicates that all expected data in the file must be modeled in the template.

`partial-template` indicates that unmatched data in the file should be ignored and passed directly through to the output. This option only works with `preserve-format`.

The @!timeout attribute

Syntax

@!timeout={minutes}@

`{minutes}` default value is 1.

Description

`timeout` represents the number of minutes that should be added onto the Configurations total timeout. A valid timeout is any integer from 0-999 (inclusive). The time-outs of all the templates in a configuration get added together, and that number is added to the default timeout for configurations (10 minutes) to get the final timeout value for the entire configuration.

The @!unix-newlines and @!windows-newlines attributes

Syntax

```
@!unix-newlines@
@!windows-newlines@
```

`unix-newlines` is the default behavior.

Description

`unix-newlines` is the default behavior and indicates that the configuration file generated by this template will have unix-style newlines (ASCII Line Feed character).

`windows-newlines` indicates that the configuration file generated by this template will have windows-style newlines (ASCII Carriage Return + Line Feed combination).

CML regular option attributes

CML attributes define and control the semantics of a CML tag. This section defines the possible option attributes you can use in a CML template. Regular options can be use either as Instruction tags or as Option attributes in other tag types.

The @! unordered-lines and @!ordered-lines attributes

Instruction tag syntax

```
@!unordered-lines@
@!ordered-lines@
```

`unordered-lines` is the default behavior.

Option attribute syntax

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;unordered-lines[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;ordered-lines[;
{option}]...]]]]@
```

Valid for groups.

Description

`unordered-lines` allows child tags of a template to appear in any order; however, position of items within ordered sequence elements is preserved. `unordered-lines` is the default behavior.

`ordered-lines` instructs the parser that child tags of the template object (lines, loops, conditionals, and so on) must appear in the file in the ordered they are specified in the template.

The unordered-elements and ordered-elements attributes

Instruction tag syntax

```
@!unordered-elements@
@!ordered-elements@
```

`unordered-elements` is the default behavior.

Option attribute syntax

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;unordered-elements[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;ordered-elements[;
{option}]...]]]]@
```

Valid for groups.

Description

`unordered-elements` allows child tags of of the current group to appear in any order; however, position of items within ordered sequence elements is preserved. `unordered-elements` is the default behavior.

`ordered-elements` instructs the parser that child tags of the group object (loops, conditionals, elements, and so on) must appear in the file in the ordered they are specified in the template.

The relaxed-whitespace and strict-whitespace attributes

Instruction tag syntax

```
@!relaxed-whitespace@
@!strict-whitespace@
```

`relaxed-whitespace` is the default behavior.

Option attribute syntax

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;relaxed-whitespace[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;strict-whitespace[;
{option}]...]]]]@
```

Valid for groups.

Description

`relaxed-whitespace` allows whitespace in the template to be matched by any combination of tabs and spaces. `relaxed-whitespace` is the default behavior.

`strict-whitespace` requires that whitespace in the template be matched exactly in the file.

The required-whitespace and optional-whitespace attributes

Instruction tag syntax

```
@!required-whitespace@
@!optional-whitespace@
```

`required-whitespace` is the default behavior.

Option attribute syntax

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;required-whitespace[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;optional-whitespace[;
{option}]...]]]]@
```

Valid for groups.

Description

`required-whitespace` requires that whitespace in the template be in the file. `optional-whitespace` makes the presence of non-significant whitespace in the file optional.

The missing-values-are-null and missing-values-are-error attributes

Instruction tag syntax

```
@!missing-values-are-null@
```

```
@!missing-values-are-error@
```

`missing-values-are-null` is the default behavior.

Option attribute syntax

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;missing-values-are-null[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;missing-values-are-error[;
{option}]...]]]]@
```

Description

`missing-values-are-null` instructs that values that are not found in the file are null, and therefore not provided in the Value Set.

`missing-values-are-error` throws an error if all values specified in a template are not found in a file or Value Set.

The case-insensitive-keywords and case-sensitive-keywords attributes

Instruction tag syntax

```
@!case-insensitive-keywords@
@!case-sensitive-keywords@
```

`case-insensitive-keywords` is the default behavior.

Option attribute syntax

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;case-insensitive-keywords[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;case-sensitive-keywords[;
{option}]...]]]]@
```

Description

`case-insensitive-keywords` match literal text in the file ignoring case. case-insensitive-keywords is the default behavior.

`case-sensitive-keywords` instructs that literal text in the template must be matched in a case-sensitive basis in the file.

The reluctant attribute

Instruction tag syntax

```
@!reluctant@
```

Option attribute syntax

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;reluctant[;{option}]...]]]]@
```

Description

`reluctant` specifies that a specific loop or sequence will try to match as few elements as possible from the configuration file. This is not the default behavior of loops and sequences.

The required and optional attributes

Instruction tag syntax

```
@!required@
@!optional@
```

`required` is the default behavior.

Using `optional` in an instruction tag may have unintended consequences.

Option attribute syntax

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;required[;{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;optional[;{option}]...]]]]@
```

Description

`required` elements must be matched (unless nested inside optional groups). `required` is the default behavior.

`optional` elements are optional.

Using `optional` as an option attribute is valid for any tag, except an instruction tag. Using `optional` in an instruction tag may have unintended consequences.

The skip-lines-without-values and show-lines-without-values attributes

Instruction tag syntax

```
@!skip-lines-without-values@
@!show-lines-without-values@
```

skip-lines-without-values is the default behavior.

Option attribute syntax

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;skip-lines-without-values[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;show-lines-without-values[;
{option}]...]]]]@
```

Description

skip-lines-without-values instructs when a line has replace elements, and all values for those elements are null, that line should be suppressed from the output. skip-lines-without-values is the default behavior.

show-lines-without-values instructs that all lines should be shown, regardless of the presence or absence of null values.

The skip-groups-without-values and show-groups-without-values attributes

Instruction tag syntax

```
@!skip-groups-without-values@
@!show-groups-without-values@
```

skip-groups-without-values is the default behavior.

Option attribute syntax

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;skip-groups-without-values[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;show-groups-without-values[;
{option}]...]]]]@
```

Description

skip-groups-without-values instructs when a group has replace elements, and all values for those elements are null, that groups should be suppressed from the output. skip-groups-without-values is the default behavior.

show-groups-without-values instructs that all groups should be shown, regardless of the presence or absence of null values.

The sequence-append, sequence-replace and sequence-prepend attributes

Instruction tag syntax

```
@!sequence-append@
@!sequence-replace@
@!sequence-prepend@
```

`sequence-append` is the default behavior.

Valid for loops and sequences.

Option attribute syntax

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;sequence-append[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;sequence-replace[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;sequence-prepend[;
{option}]...]]]]@
```

Description

`sequence-append` sequence elements child scopes are appended to sequence elements in parent scopes. `sequence-append` is the default behavior.

`sequence-replace` indicates that sequence elements child scopes replace sequence elements in parent scopes.

`sequence-prepend` sequence elements child scopes are prepended to sequence elements in parent scopes.

The not-primary-field and primary-field attributes

Instruction tag syntax

```
@!not-primary-field@
@!primary-field@
```

`not-primary-field` is the default behavior.

Option attribute syntax

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;not-primary-field[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;primary-field[;
{option}]...]]]]@
```

Description

`not-primary-field` indicates this field should not be used for the purposes of identifying duplicate items when performing list aggregation.

`not-primary-field` is the default behavior.

`primary-field` indicates this field should be used for the purposes of identifying duplicate items when performing list aggregation.

Valid for sequence and replace tags inside a sequence.

The namespace attribute

Instruction tag syntax

```
@!namespace={namespace}@
```

The default value for `{namespace}` is “/” (the root name space).

Option attribute syntax

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;namespace={namespace}[;
{option}]...]]]]@
```

The default value for `{namespace}` is “/” (the root name space).

Description

`namespace` is a string that identifies the name space within which elements with unqualified names (names without a preceding slash or period) will be stored.

The default value for `{namespace}` is the root name space, represented by the string “/” (forward-slash).

The name space value is a path name. It must start with a slash (/).

The boolean-no-format attribute

Instruction tag syntax

```
@!boolean-no-format={string}@
```

The default value for {string} is “no”

Option attribute syntax

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;boolean-no-format={string}[;
{option}]...]]]]@
```

The default value for {string} is “no”

Description

`boolean-no-format` identifies the string that will be used to match false Boolean elements. Valid for Boolean replace tags.

The boolean-yes-format attribute

Instruction tag syntax

```
@!boolean-yes-format={string}@
```

The default value for `{string}` is “yes”

Option attribute syntax

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;boolean-yes-format={string}[;
{option}]...]]]]@
```

The default value for `{string}` is “yes”

Description

`boolean-yes-format` and `boolean-no-format` identifies the strings that will be used to match boolean elements. The default value for `{string}` is “yes”

Valid for boolean replace tags.

The delimiter attribute

```
whitespace-delimited
comma-delimited
semicolon-delimited
tab-delimited
quote-delimited
delimiter
```

Instruction tag syntax

```
@!whitespace-delimited@
@!comma-delimited@
@!semicolon-delimited@
@!tab-delimited@
@!quote-delimited@
@!delimiter={string}@
```

`whitespace-delimited` is the default behavior

Option attribute syntax

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;whitespace-delimited[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;comma-delimited[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;semicolon-delimited[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;tab-delimited[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;quote-delimited[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;delimiter={string}[;
{option}]...]]]]@
```

`whitespace-delimited` is the default behavior.

Description

`delimiter` sets the default delimiter character. If not explicitly specified, sequence-delimiter and field-delimiter will inherit this value.

Valid for `replace` and `sequence` tags.

The line-comment attributes

```
line-comment-is-comma
line-comment-is-semicolon
line-comment-is-tab
line-comment-is-whitespace
line-comment
```

Instruction tag syntax

```
@!line-comment-is-comma@
@!line-comment-is-semicolon@
@!line-comment-is-tab@
@!line-comment-is-whitespace@
@!line-comment={string}@
```

There is no default value for `{string}`.

Option attribute syntax

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;line-comment-is-comma[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;line-comment-is-semicolon[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;line-comment-is-tab[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;line-comment-is-whitespace[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;line-comment={string}[;
{option}]...]]]]@
```

There is no default value for `{string}`.

Description

`line-comment` sets the character that indicates that the remainder of the line will be parsed as a comment.

The sequence-delimiter attribute

```
sequence-delimiter-is-comma
sequence-delimiter-is-semicolon
sequence-delimiter-is-tab
sequence-delimiter-is-whitespace
sequence-delimiter-is-quote
sequence-delimiter
```

Instruction tag syntax

```
@!sequence-delimiter-is-comma@
@!sequence-delimiter-is-semicolon@
@!sequence-delimiter-is-tab@
@!sequence-delimiter-is-whitespace@
@!sequence-delimiter-is-quote@
@!sequence-delimiter={string}@
```

By default, `sequence-delimiter` uses the value of delimiter. The default for the latter is `whitespace-delimited`.

Option attribute syntax

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;sequence-delimiter-is-comma[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;sequence-delimiter-
issemicolon[;{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;sequence-delimiter-is-tab[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;sequence-delimiter-is-
whitespace[;{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;sequence-delimiter-is-quote[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;sequence-delimiter={string}[;
{option}]...]]]]@
```

By default, `sequence-delimiter` uses the value of delimiter. The default for the latter is `whitespace-delimited`.

Description

`sequence-delimiter` sets the character that separates items within a sequence. By default, `sequence-delimiter` uses the value of delimiter. The default for the latter is `whitespace-delimited`.

Valid for sequences.

The field-delimiter attribute

```
field-delimiter-is-comma
field-delimiter-is-semicolon
field-delimiter-is-tab
field-delimiter-is-eol
field-delimiter-is-whitespace
field-delimiter-is-quote
field-delimiter
```

Instruction tag syntax

```
@!field-delimiter-is-comma@
@!field-delimiter-is-semicolon@
@!field-delimiter-is-tab@
@!field-delimiter-is-whitespace@
@!field-delimiter-is-quote@
@!field-delimiter={string}@
```

By default, `field-delimiter` uses the value of delimiter. The default for the latter is `whitespace-delimited`.

Option attribute syntax

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;field-delimiter-is-comma[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;field-delimiter-is-semicolon
[;{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;field-delimiter-is-tab[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;field-delimiter-is-whitespace
[;{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;field-delimiter-is-quote[;
{option}]...]]]]@
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;field-delimiter={string}[;
{option}]...]]]]@
```

By default, `field-delimiter` uses the value of delimiter. The default for the latter is `whitespace-delimited`.

Description

`field-delimiter` sets a character that will be used to terminate parsing for a replace element value. By default, `field-delimiter` uses the value of delimiter. The default for the latter is `whitespace-delimited`.

Valid for replace tags and sequence tags.

The line-continuation attribute

Instruction tag syntax

```
@!line-continuation={string}@
```

Option attribute syntax

```
@[{level}]{tag-type}[[{source}][;[{type}][;[{range}][;line-continuation={string}[;
{option}]...]]]]@
```

Description

`line-continuation` sets a character that will be used to indicate that the current line in a config file should be wrapped to the subsequent line.

Use DTD tags in CML

CML supports Document Type Definition (DTD) tags that can be used to pre-define attributes for a CML tag. Using a DTD tag in CML allows you to change some aspects of how the template is displayed in the SA Client. The DTD definition generally goes in the beginning of a file and the tag gets shortened to just a name and a tag type.

The main advantage of using DTD tags in CML is the ability to define 'printable' and 'description' values, which are reflected in the SA Client, improving usability. DTD definitions can be used to define any tag that has a name; for example loop tags, loop target tags, replace tags, and so on, but not tags like instruction tags or block tags. DTD tags in CML are also inherently multi-line tags.

Example of DTD tags

Here we will take a tag and create a DTD version of that tag. A DTD tag in CML is not that different than a regular CML tag; it contains all the elements of a tag minus the “tag type.”

For example, in the CML tag below:

```
@*deny_header;unordered-string-set;;sequence-delimiter=":";optional@
```

this is an instance representing the following format in CML:

```
@<tag type><name>;<data type>;;<option1>;<option2>@
```

The DTD version of this takes the existing elements and reorders them as follows:

```
<start code block>
@~<name>
type = <data type>
description = <description>
printable = <printable>
<option1>
<option2>
...
@
@<tag type><name>@
<end code block>
```

As you can see, this usage also allows for the addition of two new elements: “description” and “printable”. Defining “printable” will define the main text for this tag in the SA Client. Defining “description” will create a description for this value in the SA Client that is viewable when you move your mouse pointer over the field in the value set editor in the SA Client.

Here is the same tag in full DTD format:

```
<start code block>
@~deny_header
type = unordered-string-set
printable = Headers to Deny
description = This is a list of headers that IIS should deny
sequence-delimiter = ":"
optional
@
```

```
@*deny_header@
<end code block>
```

There are a couple things to notice in the example above. In defining a value for “description,” the value can span multiple lines, as long as the lines following the first line have whitespace as the first character.

Options go on a line by themselves, where you have `<option>=<value>` you need to insert spaces before and after the `"="` sign.

Now, where ever you use the tag `@*deny_header@`, the parser will use the predefined DTD for all that tags' information.

Redefining a DTD defined tag, `@*deny_header@`, by using a line like `@*deny_header;unordered-string-set@` will cause the CML template to become invalid.

> **Note:**
> Note also that DTD style CML is not currently required, but is most obvious when viewing the Application Configuration the SA Client. If you don't use DTD tags you will not see the 'printable' and 'description' fields, instead you will only see the underlying variable name.

Sequence aggregation

Because Application Configuration values can be set across many different levels in the Application Configuration inheritance hierarchy (also referred to as the inheritance *scope*), it is important that you be able control the way multiple sequence values are merged together when you push an Application Configuration on to a server.

ACM allows you to control the way sequence values are merged across inheritance scopes. This means that you can, for example, add some values to a sequence in the Customer scope, Group scope, and the Server scope, and all the values will be merged together to form the final sequence.

The manner in which sequence values are merged is controlled by special tags in the CML template, using three different sequence merge modes:

- "Sequence replace " on the next page: Sequence values from more specific scopes completely replace those from less specific scopes. This occurs for both sequences of sets and lists.
- "Sequence append " on page 419: For lists, values at more general scopes are *appended* (placed after) to those at more specific scopes. Duplicates, if present, are not removed. For sets, the behavior is the same, except duplicates are merged. For lists, duplicates are identified according to child elements marked with the `primary-key` tag, and then merged. For scalars, this is done by

simply removing duplicate values, leaving only the value from the most specific scope (the last occurrence is the merged sequence). This is the default mode, and will be used if nothing else is specified.

- **Sequence Prepend**: Works the same as append, but values at more general scopes are *prepended* (placed before) to those at more specific scopes.

For example, with these two sets:

- “a, b” – At a more specific (inner) level of the inheritance scope, for example, server instance level.
- “c, d” – At a more general (outer) of the inheritance scope, for example, the server group level.

When the application configuration template is pushed onto the server, the merging results would be:

- Sequence replace: “a, b”
- Sequence append: “a, b, c, d”
- Sequence prepend: “c, d, a, b”

Sequence aggregation occurs not only between scopes, but also within a scope itself. This is evident if there are duplicate values within a sequence of name spaces.

Sequence replace

In the Replace merge mode (CML tag “`sequence-replace`”), the contents of a sequence defined at a particular scope replace those of less specific scopes, and no merging is performed on the individual elements of the sequence.

For example, if the `sequence-replace` tag has been set for a list in an configuration template CML source, then values set for that list at the server instance level will override, or replace, those set at the group level and at the Application Configuration default values level.

For example, if a list in an `etc/hosts` file was defined at the group level (outer) as the following:

```
/system/dns/host/1/ip                      127.0.0.1
/system/dns/host/1/hostnames/1  localhost
/system/dns/host/1/hostnames/2  mymachine
/system/dns/host/2/ip           10.10.10.10
/system/dns/host/2/hostnames/1  loghost
```

And the same list was defined at the device scope (inner), as the following:

```
/system/dns/host/1/ip           127.0.0.1
/system/dns/host/1/hostnames/1  localhost
/system/dns/host/1/hostnames/2  mymachine.mydomain.net
/system/dns/host/2/ip           10.10.10.100
/system/dns/host/2/hostnames/1  mailserver
```

If template had defined the `/system/dns/host` element with the `sequence-replace` tag, the final results of the configuration file on the server after the push would be:

```
127.0.0.1 localhost mymachine.mydomain.net
10.10.10.100 mailserver
```

Sequence append

When the append list merge mode (CML tag “`sequence-append`”) is used for sequences, the values at more general scopes are appended (placed after) those of more specific scopes. Sequence append mode is the default mode for merging list values. If nothing is specified in the CML of the template, the sequence append will be used.

If a list in an `etc/hosts` file was defined at the group level (outer) as the following:

```
/system/dns/host/1/ip                      127.0.0.1
/system/dns/host/1/hostnames/1  localhost
/system/dns/host/1/hostnames/2  mymachine
/system/dns/host/2/ip           10.10.10.10
/system/dns/host/2/hostnames/1  loghost
```

And the same list was defined at the device scope (inner), as the following:

```
/system/dns/host/1/ip           127.0.0.1
/system/dns/host/1/hostnames/1  localhost
/system/dns/host/1/hostnames/2  mymachine.mydomain.net
/system/dns/host/2/ip           10.10.10.100
/system/dns/host/2/hostnames/1  mailserver
```

Using the value sets from the above example, if the `/system/dns/host` element was a list with the `sequence-append` tag set in the configuration template, the final results of the configuration file on the server after the push would be:

```
127.0.0.1 localhost mymachine.mydomain.net
10.10.10.100 mailserver
```

```
127.0.0.1 localhost mymachine
10.10.10.10 loghost
```

But since it is not allowable for a hosts file to contain duplicate entries, the`/system/dns/host` element will have to be flagged in the configuration template as a set rather than a list, because sets do not allow duplicates. To avoid duplication of the list values in the example, the configuration template author would use the Primary Key option.

Primary key option in sequence merging

When operating in append mode on sets, new values in more specific scopes are appended to those of less specific ones, and duplicate values are merged with the resulting value placed in the resulting sequence according to its position in the more specific scope.

How this affects merged sequence values depends on what kind of data is contained in the sequence:

- For elements in a sequence which are scalars, the value from the most specific scope is used. In other words, values at the server instance level would replace the values at the group level.
- For elements which are namespace sequences, the value is obtained by applying the merge mode specified for that element (in this example, append) based upon matching up the primary fields.

To avoid the duplication of the `/system/dns/host/.ip` value, the configuration template author would use the CML `primary-key` option. With this option set, ACM will treat entries with the same value for `/system/dns/host/.ip` as the same and merge their contents.

In the example above, the final results of the configuration file on the server after the push would be:

```
127.0.0.1 localhost mymachine.mydomain.net mymachine
10.10.10.100 mailserver
10.10.10.10 loghost
```

> **Note:**
> Since it is possible to have a set without primary keys, if there are scalars in the sequence, then an aggregation of all scalar values will be used as the primary key. If there are no scalars, then the aggregation of all values in the first sequence will be used as the primary key. Although this is an estimate, in most cases the values will be merged effectively. To ensure that the correct values are used as primary keys, we recommend that you always explicitly set the primary key in a sequence.

Sequence prepend

When the prepend list merge mode (CML tag “`sequence-prepend`”) is used for sequences, the values at more general scopes are prepended (placed before) those of more specific scopes.

For example, if a sequence in an `etc/hosts` file was defined at the group level (outer) as the following:

```
/system/dns/host/1/ip 127.0.0.1
/system/dns/host/1/hostnames/1 localhost
/system/dns/host/1/hostnames/2 mymachine
/system/dns/host/2/ip 10.10.10.10
/system/dns/host/2/hostnames/1 loghost
```

And the same sequence was defined at the device scope (inner), as the following:

```
/system/dns/host/1/ip 127.0.0.1
/system/dns/host/1/hostnames/1 localhost
/system/dns/host/1/hostnames/2 mymachine.mydomain.net
/system/dns/host/2/ip 10.10.10.100
/system/dns/host/2/hostnames/1 mailserver
```

If the `/system/dns/host` element was a set with the `sequence-prepend` tag set in the configuration template, the final results of the configuration file on the server after the push would be:

```
10.10.10.10 loghost
127.0.0.1 mymachine localhost mymachine.mydomain.net
10.10.10.100 mailserver
```

CML grammar

The following table describes CML grammar illustrating several types of CML tags.

CML grammar

CML tag/element	Description
replace-tag	"@" source [";" [type] [";" [range] *option]] "@"
data-definition-tag	"@~" source CRLF *def-line "@"
conditional-tag	"@" [group-level] "?" source [";" [type] [";" [range] *option]] "@"
loop-tag	“@" [group-level] "*" source [";" [type] [";" [range] *option]] "@"
loop-target-tag	"@.@"
block-tag	"@" [group-level] "[" *option "@"

CML grammar, continued

CML tag/element	Description
block-termination-tag	"@" [group-level] "]@"
line-continuation-tag	"@\"
instruction-tag	"@!" *option "@"
single-line-comment	"@#" [string CRLF]
multi-line-comment	"@##" *[string / CRLF] "#@"
def-line	type-line / range-line / option-line / printable-line / desc-line
type-line	"type" WSP "=" WSP type-elem CRLF
range-line	"range" WSP "=" WSP range CRLF
option-line	option-elem CRLF
printable-line	"printable" WSP "=" WSP string CRLF
desc-line	"description" WSP "=" *[WSP string CRLF]
group-level	int
source	absolute-path / relative-path / local-path
absolute-path	"/" path-component* name
relative-path	[path-component*] name
path-component	(name / sequence-id) "/"
sequence-id	int
local-path	"." name
name	string
type	sequence / type-elem
sequence	[order "-"] type-elem "-" sequence-elem
sequence-elem	"set" / "list"
type-elem	"int" / "string" / "ip" / "port" / "file" / etc...
order	ordered" / "unordered"
range	and-range *["," and-range]
and-range	range-elem *["&" range-elem]
range-elem	numeric-range / string range

CML grammar, continued

CML tag/element	Description
numeric-range	gt-range / ge-range / lt-range / le-range / eq-range
string range	string-literal / regular-exp
gr-range	int ">"
ge-range	int ">="
lt-range	">" int
le-range	">=" int
eq-range	"=" int
string-literal	<"> string <">
regular-exp	"r" <"> string <">
option	";" option-elem
option-elem	option-name / option-nv
option-nv	option-nv
option-name	string
option-value	string

Application deployment

- **Application experts** understand precisely how their multi-tiered applications work and how the various parts of the application (for example, code and scripts) should be deployed for optimum results.
- **Operations experts** understand the hardware in the data center and how it is used to run multi-tiered applications most effectively.
- **Environment owners** are responsible for the hardware and software in a specific deployment environment (for example: Development, QA, User Acceptance Testing, Pre-Production, or Production).
- **Deployment specialists** are responsible for deploying applications into environments in the pertinent lifecycle.
- **Application Deployment administrators** are responsible for specifying the environments, tiers, lifecycles, scripts, and the Code component source types that can be used to deploy applications.

The following stakeholders also benefit by understanding howApplication Deployment works:

- **SA administrators** are responsible for all HPE Server Automationadministration tasks. They control the privileges and permissions available to each user role, and they decide which servers are managed by SA. They may also be responsible for installing and updating SA.
- **Development team** members and managers design and implement applications that are eventually deployed in the Production environment.
- **QA team** members and managers must understand how applications are flowing through the lifecycle and how the QA environment is affected. They can use Application Deployment to deploy and test applications.

Prerequisites

Application Deployment is part of Server Automation (SA). Refer to the SA Installation Guide for information about prerequisites for SA.

To perform the various operations described in this guide, you must have the proper permissions. See "Setting permissions" on page 579 for more information.

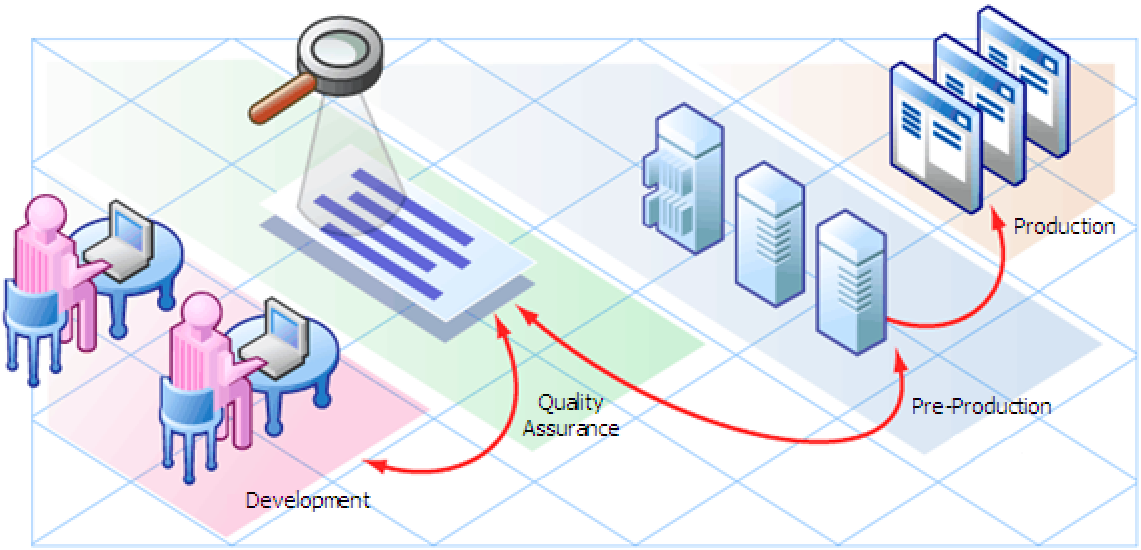
The Application Deployment user interface requires Adobe Flash Player version 9.0 (or later).

Overview

Why automated Application Deployment is important

Creating, deploying, and testing custom applications has become a complex process that warrants a software development lifecycle. The lifecycle provides structure for the various activities involved. Here is an example of a simple software development lifecycle:

Typical software development lifecycle



In this software development lifecycle, an application will move from Development to Quality Assurance (QA). QA tests the application, and often there are several cycles of testing and fixes between QA and development. After QA approves the application, it may be deployed—either for the first time or as an upgrade—to verify that the application works in the pre-production environment (which is very similar to the Production environment). This deployment may be used to carry out performance testing. Only when this is successful is the application moved into production.

When performed manually, this process is time-consuming and prone to error. Challenges arise from fragmentation of both people and processes. For example, applications may require components from geographically dispersed development teams; QA may control the QA environment, while system administrators are responsible for the pre-production and production environments. This requires a great deal of communication and coordination.

Some additional challenges are:

- Complex environments complicate the process.
- There is typically not a consistent way for Development to describe the steps necessary to deploy a release.
- QA must have the machines necessary to create environments. Then, they must create and clean up these environments repeatedly.
- They may even have to perform multiple deployments to bring the application up to the latest production level.
- A product may work in QA but fail in pre-production due to inconsistent processes used in the two environments.
- The Pre-Production environment must precisely mimic the Production environment.
- Each time the code changes, software engineers must re-package the code.
- Someone must be responsible for the Production machines and knowing what needs to be updated.
- There is no single place for release managers and the teams that they work with to communicate the current state of a release.
- Lack of auditing, reporting, and metrics lead to a lack of process improvement, because organizations do not know what needs to be fixed.
- Security issues arise from conflicting needs between Development and Operations.

Application Deployment in HPE Server Automation reduces the complex communications necessary to deploy applications by providing a single point of access where everyone involved can view or enter data that is relevant to them and to their role.

Application Deployment also integrates with other HPE Software technologies to ensure that applications are successfully moved into Production.

Application architecture

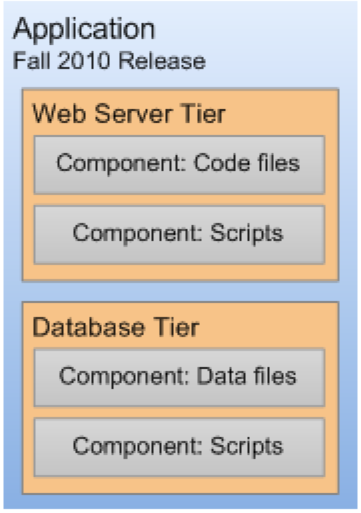
Applications are made up of a variety of **components**, including the following:

- Code components (for example: HTML files, JAR files, WAR files, .NET assemblies, or databases)

- Scripts
- Configuration files
- SA Application Configurations
- SA Software Policies
- Packages from the SA library
- HPE Operations Orchestration (OO) flows
- Windows Registry settings

These components are deployed to the appropriate tiers, such as web servers, application servers, and databases.

Simple application with two tiers



A single application can have multiple **releases**. A release, in turn, typically has multiple **versions**. A version is an immutable “snapshot” of a release at a specific point in time. When you deploy an application to an environment, you deploy a specific version.

When you specify the components included in a release, you can specify both **rollback** and **undeploy** instructions for each component.

When rollback is enabled, Application Deployment performs a backup before deploying. In the event that a failure occurs during deployment, the rollback instructions are used to return the target to its previous state. In this situation, Application Deployment will retrace its steps and undo each step of the deployment according to the rollback instructions.

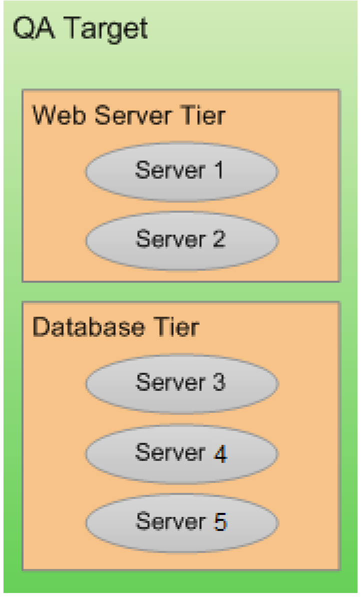
If undeploy is enabled, the undeploy instructions are used to uninstall the application.

You can create application **groups** to help you organize your applications in the most convenient manner. This is an organizational tool that is provided strictly for your benefit; it does not affect the deployment of applications.

Operations architecture

On the operations side, you have **targets** where you deploy applications. Each target is generally suited to support one or more applications, depending on its structure. A target is structured by use of **tiers**—for example, an application server tier or a database tier. Each target is associated with an **environment**, such as QA or Production. After you have created your environments, you can see how they map to your software development **lifecycle**.

Target with two tiers



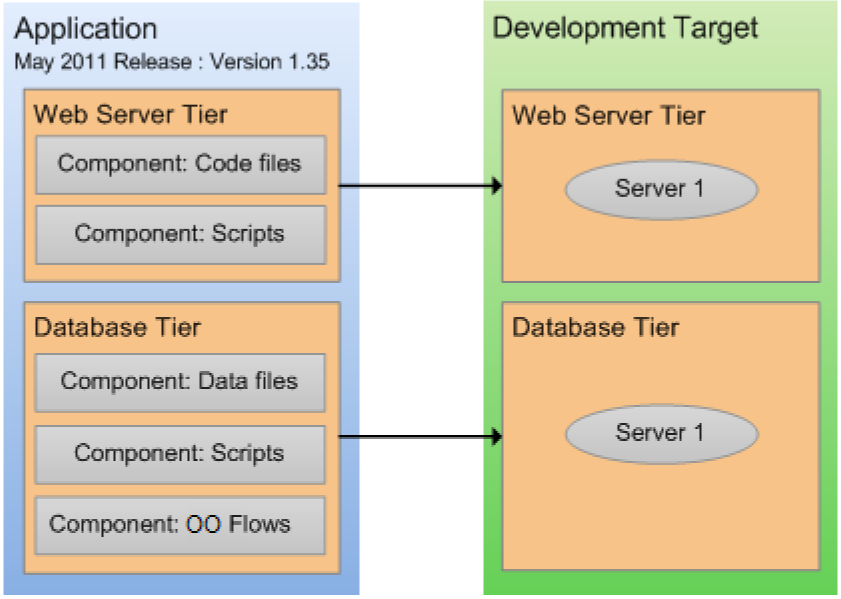
How the application and operations architectures align

Tiers are the mechanism used to connect applications to targets. An application is deployed by mapping the components that are part of each tier to the target servers that belong to the same tier. Regardless of the size or composition of the targets, the application tiers do not change.

Consider the following example of how a simple application—perhaps a patch—progresses from the Development environment, through the QA environment, and ultimately to the Production environment.

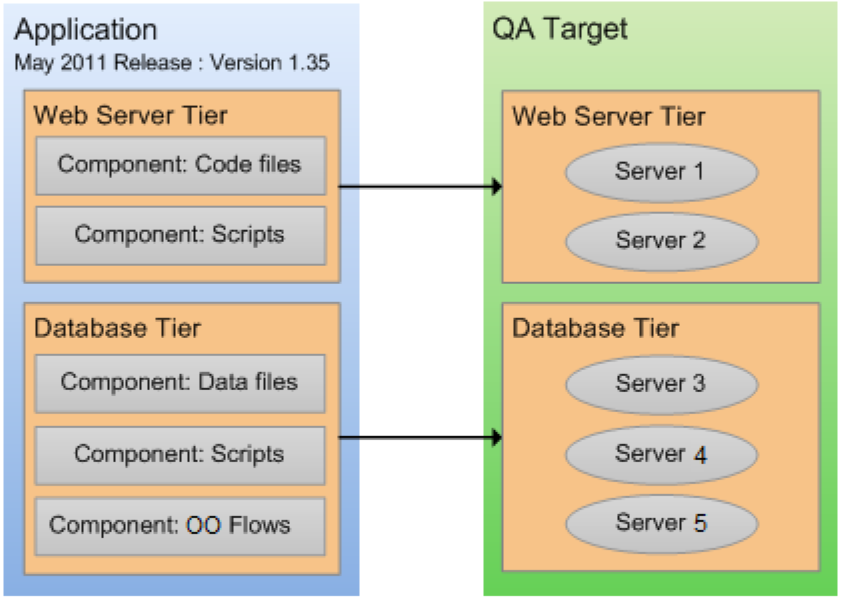
During the development phase, the application changes frequently, so many versions are deployed to a small testing environment—in this case, the same server runs both the web server and the database.

Deploying to a small development target



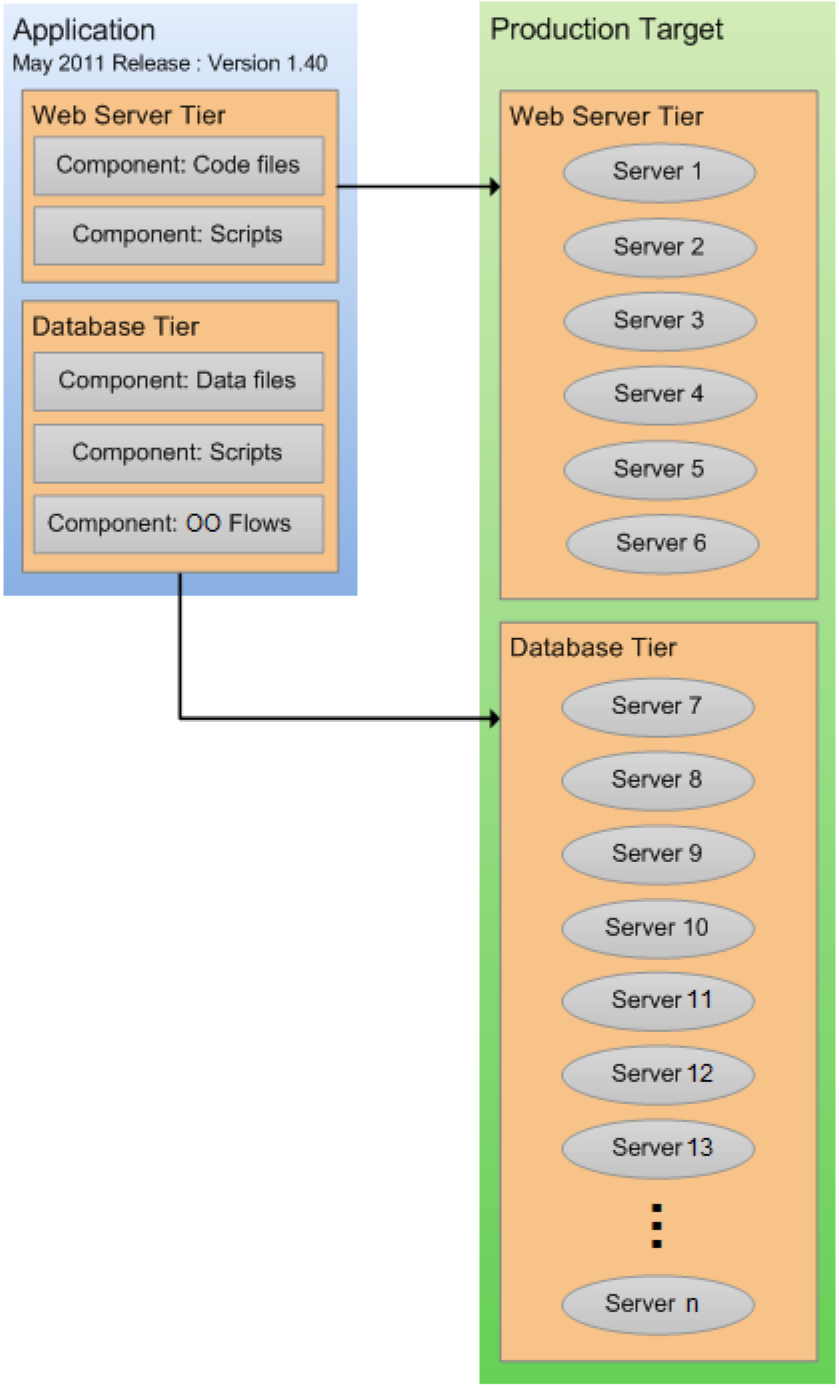
When the application is promoted into the QA environment, there are fewer iterations. The environment is slightly larger. During the QA phase, the application is likely deployed in both the Development and the QA environment as problems are detected and fixed.

Deploying to a slightly larger QA target



Eventually, when the application is stable and thoroughly tested, it is promoted to the Production environment. This is usually a much larger environment, as shown in figure above.

Deploying to a production environment



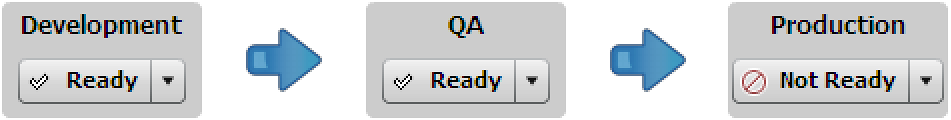
Note that in this example, the production database is hosted on a server cluster, such as an Oracle Real Application Cluster (RAC).

How an application moves through the software lifecycle

Application Deployment enables you to create customized lifecycles so that you can exercise finely grained control over an application as it progresses from the Development phase through full deployment in a Production environment.

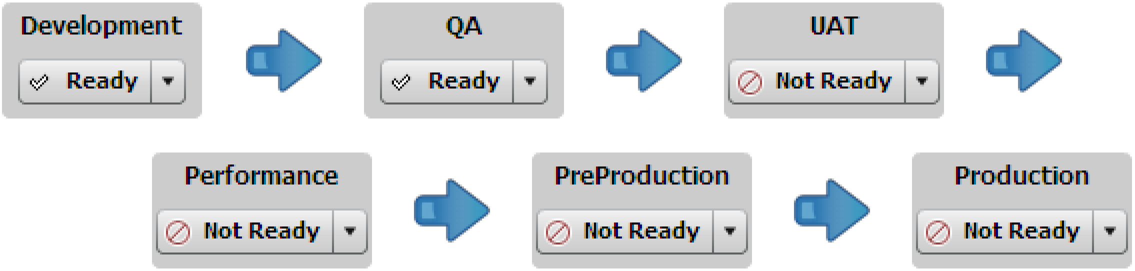
A lifecycle consists of a series of environments. An environment consists of a group of servers. You can use different lifecycles for different purposes. For a very small application or minor patch, for example, you might use a simple lifecycle like this:

Patch lifecycle



For a more complex application—such as a new product or major upgrade—you would use a more sophisticated lifecycle like this one:

Complete lifecycle



Through the use of lifecycles, the Application Deployment helps ensure a seamless transition between application development and operations.

In the very beginning of the lifecycle, an application typically undergoes frequent changes. You might, for example, create a new version for each nightly build and deploy it to the servers in the Development environment.

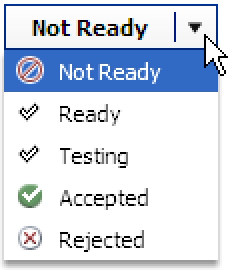
As the application becomes more stable, it becomes ready for formal QA testing. At this point, you might install the latest build on each server in the QA environment twice per week.

As the application moves forward in the lifecycle, changes become less frequent, and fewer deployments are required. Eventually, there will be a small number of release candidates that are deployed for user acceptance testing (UAT) and performance testing.

Upon successfully passing UAT and demonstrating acceptable performance, the release candidate could be deployed to Pre-Production and, ultimately, to Production.

SA determines which environments are included in the various lifecycles, according to the standards at your company.

Environment status settings



Each environment has a certain status. Application Deployment will not allow you to deploy to environments whose status is Not Ready—in this case, no targets in the environment can be selected.

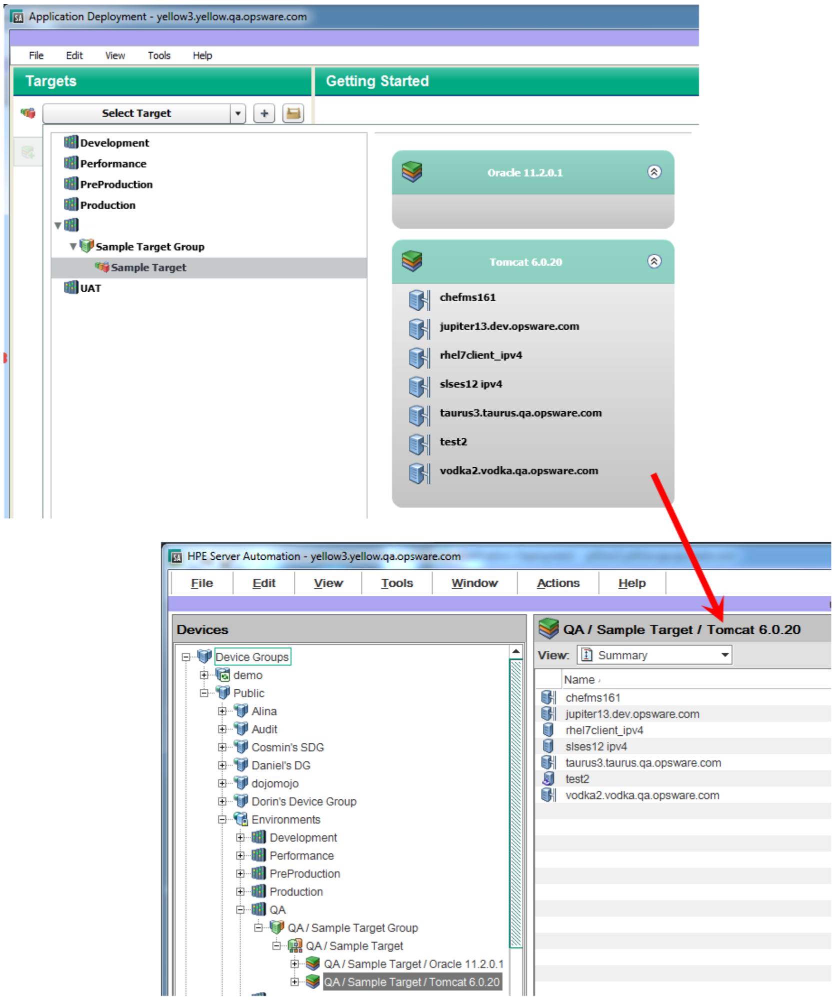
You can use the status field to control where an application can be deployed. The initial setting for each environment is set by the Application Deployment administrator. If a user has permission to deploy to a particular environment, that user can change the environment’s status at deployment time.

Your Application Deployment administrator determines which users have permission to deploy applications into which environments. See "Setting permissions" on page 579 for details.

Environments and device groups

Environments in the Application Deployment context are mirrored as device groups in SA:

Application Deployment and SA views



Name
chefms161
jupiter13.dev.opsware.com
rhel7client_ipv4
slses12 ipv4
taurus3.taurus.qa.opsware.com
test2
vodka2.vodka.qa.opsware.com

The Application Deployment environments are located under the Environments device group on the Devices screen. These device groups are read-only (cannot be modified by users), and they are managed by Application Deployment.

Each device group under Environments has a special icon that signifies what type of Application Deployment object it is:

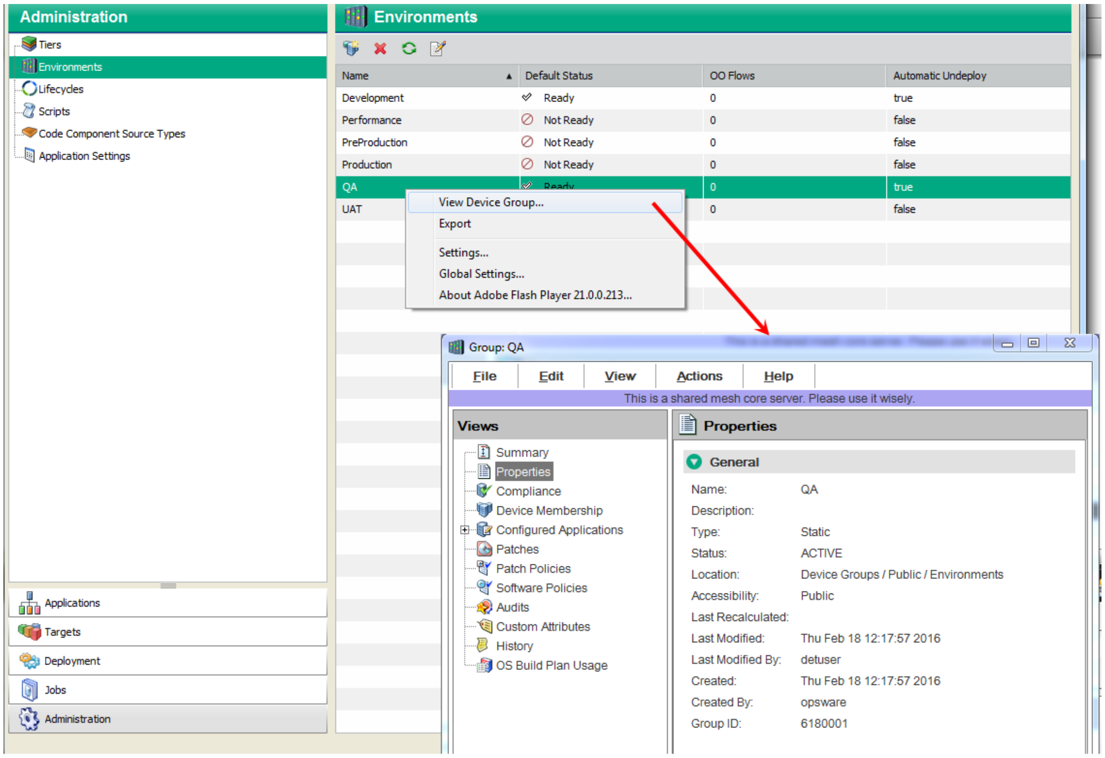
- Environment
- Target
- Tier

Target and tier device group names are paths indicating their parent environment and target, respectively.

When you make a change to an environment or target in the Application Deployment user interface—for example, renaming a target—that change is also made in the corresponding Device Group. This can take time, during which you may see a “synchronizing” message. You may need to perform a refresh in the main SA user interface to see the change.

The environment owner or Application Deployment administrator can use the right-click shortcut menu to open a Device Group window or force an immediate synchronization, as shown here (see "Administering environments" on page 596).

Device group Window for an environment



Name	Default Status	OO Flows	Automatic Undeploy
Development	✓ Ready	0	true
Performance	⊘ Not Ready	0	false
PreProduction	⊘ Not Ready	0	false
Production	⊘ Not Ready	0	false
QA	✓ Ready	0	true
UAT		0	false

This is a shared mesh core server. Please use it wisely.

Name: QA
Description:
Type: Static
Status: ACTIVE
Location: Device Groups / Public / Environments
Accessibility: Public
Last Recalculated:
Last Modified: Thu Feb 18 12:17:57 2016
Last Modified By: detuser
Created: Thu Feb 18 12:17:57 2016
Created By: opsware
Group ID: 6180001

Example: Preparing an environment and deploying an application

Imagine that you are opening an online store. You want to make the site available to the public as soon as possible. The application that handles orders is still in Development, but it is almost ready to move to QA.

Your application has two tiers: an Apache 2.2.9 web server and an Oracle 11g database. It consists of several components that are stored in your source code control system.

Here are the steps that your team would follow to use Application Deployment to move the new application into the QA environment for testing:

1. Your Application Deployment administrator sets the permissions required. For example:
   - The Application Expert needs permission to Edit the application and Deploy it into the Development environment.
   - The Deployment Specialist needs permission to View (but not Edit) the application and Deploy it into the QA environment.
2. Your Application Expert defines the application architecture by creating the pertinent components and assigning them to a Web Server tier and a Database tier.
3. Your QA Environment Owner then reviews the application architecture and prepares the QA environment needed for this application.
   a. First, he creates a target, called Store Orders. The target must contain the same tiers that were specified in the application architecture.
   b. Next, he specifies the servers that belong to each tier—the servers must already be managed by SA. The QA environment for the online store is now ready.
4. When your Software Engineering team decides that the first version of the application is ready for testing, they create a version in preparation for deployment. Next, they inform QA that the application is ready for testing.
5. A Deployment Specialist then deploys the application to the QA environment, and the QA engineers begin testing.

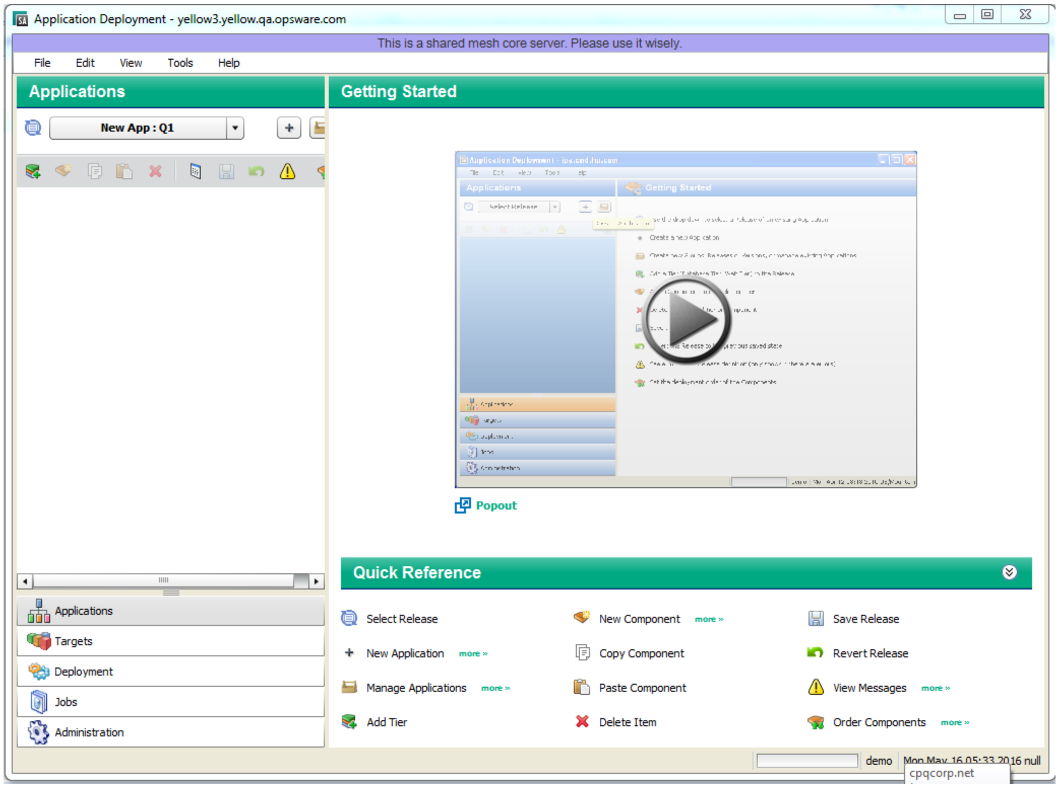
Your goal is for the team to move through the testing process with the minimum number of iterations between Development and QA. Automating the deployment reduces the number of iterations.

Once the application completes the QA cycle, moves to and passes the pre-production phase, and finally moves to the production phase, you can open your store.

About the Application Deployment user interface

When you open the Application Deployment tool for the first time—and whenever you create a new application—the user interface looks like this:

Application Deployment interface – Initial view



This is a shared mesh core server. Please use it wisely.

In this initial view, there are two tools that you can use to learn more about the Application Deployment user interface:

- The Getting Started area contains a short video demo that demonstrates the function of this screen.

  Click the ▶ button to start the video. To play it in a separate window, click Popout.

- The Quick Reference area lists the functions of the tool bar buttons available on this screen.

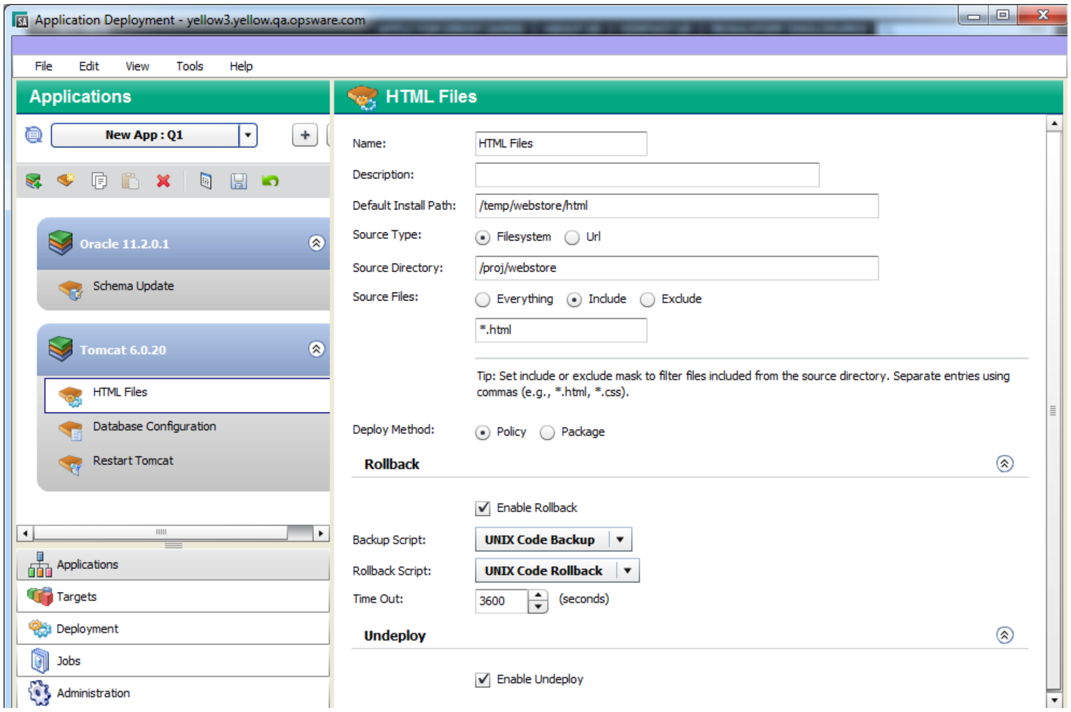
  Hover over each function to view a tooltip. Click the more>> links to open related online help topics in a separate browser window.

The buttons in the lower left corner open the different functional areas of the Application Deployment user interface. The button that is highlighted shows you the area in which you are working. Before you move to a different area, you must either save or revert any changes that you have made.

> **Note:** Application Deployment comes with a sample application and sample target to help you get started.

After you add tiers to your application and create components within the tiers, the interface looks like this:

Application Deployment interface – Application with two tiers



Name: HTML Files

Description:

Default Install Path: /temp/webstore/html

Source Type: ● Filesystem ○ Url

Source Directory: /proj/webstore

Source Files: ○ Everything ● Include ○ Exclude

*.html

Tip: Set include or exclude mask to filter files included from the source directory. Separate entries using commas (e.g., *.html, *.css).

Deploy Method: ● Policy ○ Package

Rollback

☑ Enable Rollback

Backup Script: UNIX Code Backup

Rollback Script: UNIX Code Rollback

Time Out: 3600 (seconds)

Undeploy

☑ Enable Undeploy

The left pane shows you the structural view and provides navigation controls.

The right pane shows you the content of the item selected on the left. In this example, the HTML Files component in the Tomcat 6.0.20 tier is selected.

Naming rules

The following rules apply to the Name property for all items:

- Names cannot begin or end with a space.
- Names cannot begin with a punctuation mark—in other words, they must begin with a letter or a number.
- Names cannot contain any of the following characters; newline, tab, slash (/), or backslash (\).

The Application Deployment user interface will not allow you to violate the naming rules. If a Name does not comply with these rules, a red box appears around the Name box.

Modifying information in a text box

When you modify information in a text box (either in the right pane or in a dialog), note the following:

- When you select a text box, a blue border appears around that box.
- If you violate any of the "Naming rules" on the previous page, a red box appears around the text box.

You can use Ctrl+C, Ctrl+X, and Ctrl+V to copy, cut, and paste information in any editable text box. You can also use the corresponding items on the Edit menu if you prefer.

Saving and reverting changes

Whenever you modify an item in the Application Deployment user interface, the Save and Revert buttons become active:

- Click the **Save** button 💾 to save your changes to the Application Deployment database.
- Click the **Revert** button ↶ to discard your changes and reload the most recently saved values from the database.

Before you can move to another screen—for example, from the Applications screen to the Targets screen—you must either save or revert your changes.

Managing applications and targets

Both the Applications screen and the Target screen have “managers” that you can use to quickly assess the current state of your applications and targets and perform various tasks. In the Application Manager, for example, you can create a new release or rename an application.

In the managers, you can select multiple items and then perform the same operation on all of them. For example, you can select several applications and then delete them.

To open a manager, click the button to the right of the drop-down list in the upper left corner.

See "Managing applications" on page 460 and "Managing targets" on page 522 for details.

Validation messages

The “Warning” sign ⚠ appears on the tool bar whenever a validation error has occurred. It also appears when an item is not yet fully configured.

Click the “Warning” sign ⚠ to learn more about the particular error that has been detected.

For example, after you create a new application, but before you have added any components, the following validation error is reported:

```
“No components in release”
```

Validation error messages also appear if you attempt to perform an operation that fails. For example, if you try to create a version for an application whose component are incomplete, messages similar to the following would be displayed.

```
“Cannot create version from this release. Please correct errors below: Script
Component ‘Sample Script Component’: No content.

Code Component ‘Sample Code component’: Source Directory must be set.

Code Component ‘Sample Code component’: Default Install Path must be set.”
```

Quick start

With the Application Deployment tool, application developers and operations teams can work seamlessly together to manage the promotion of mission-critical business applications from engineering into production. You can use Application Deployment to define and deploy complex multi-tier Java EE and .NET applications.

To run the Application Deployment tool from the SA Client, select **Tools > Application Deployment**.

When defining applications, you can:

- Quickly create application definitions and work with multiple simultaneous releases.
- Define the application’s structure using tiers.
- Use software policies to ensure that middleware required by your application is already in place or deployed along with your application.

Control application content using new component types, software library objects, and (optionally) HPE Operations Orchestration (OO) flows.

Specify application default parameters for OO flows and other component types. These parameters can be adjusted, if appropriate, at deployment time to match different environments—for example, appropriate database connection settings may differ for the application between QA and production.

When defining targets, you can:

Define a collection of servers, instances, and databases commonly intended to function together to support a particular application.

Structure targets using the same collection of tiers defined in the application.

Use software policies to ensure that middleware required by your application is already in place or deployed along with your application.

Populate target tiers with managed servers, instances, and databases.

Associate an OO flow with a target tier.

>This Quick Start shows you how to create and deploy a simple application:

"Overview" below

"Quick start" on the previous page

"Quick start" on the previous page

"Quick start" on the previous page

"Quick start" on the previous page

This Quick Start assumes you have installed Server Automation (SA) and can run the SA Client. For more information, see the SA 10.50 Install Guide.

Overview

Application Deployment automates the process of deploying applications. This Quick Start tutorial provides the information you will need to get started quickly.

The basic steps are as follows:

	An application has one or more **releases**. Each release is organized in **tiers**—for example, an application server tier, a web server tier, and a database tier. Each tier includes one or more **components**. These represent the content of the application and implement its functions. See Step 1: Define your application – Tiers and components.
	You create a **version** of a release and deploy that version. Each version is an immutable snapshot of your release. A version is immutable to ensure that the same components that were tested in QA are being

	deployed to production. See Step 2: Create a version to deploy.
	Targets are groups of **servers** that host the **tiers** of your application—such as a database tier, an application server tier, or a web server tier. A particular tier may run on multiple servers. See Step 3: Define Your Targets.
	The components in each tier of your application are deployed to selected targets in your environment that have those tiers. See "Step 4: Deploy a version to a target" on page 452

Click each of the links above for basic instructions. The subsequent sections in this book provide more detailed information related to these tasks.

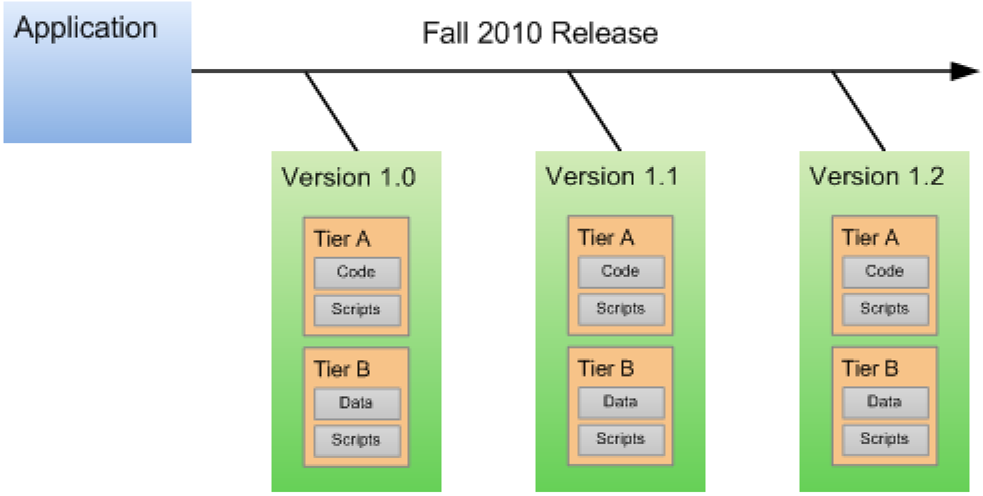
Step 1: Define your application – Tiers and components

An **application** consists of one or more **tiers**. Each tier includes **components**, which implement the functionality of the application.

Applications exist in the form of one or more **releases**. Each release can have multiple **versions**. When you deploy your application, you create a version that consists of the components for each tier. A version is an immutable “snapshot” of a release at a particular point in time.

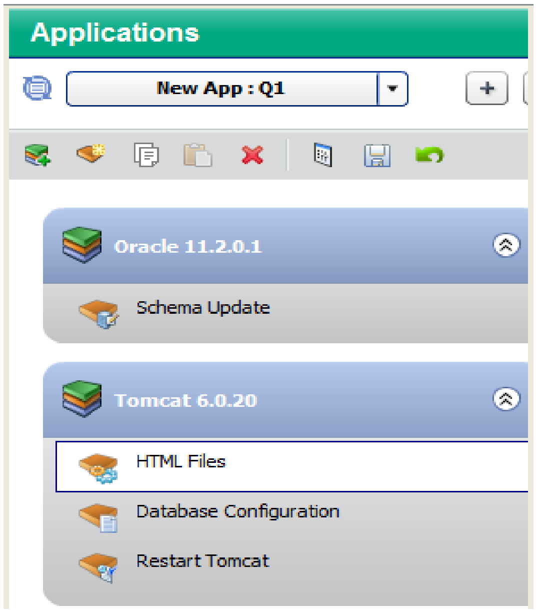
The following diagram shows an application made up of 2 tiers that has three versions of the Fall 2010 Release.

How applications are organized



The following application, for example, has an Oracle 11.2.0.1 database tier and a Tomcat 6.0.20 application server tier. The components in the Tomcat tier are an HTML file, a configuration file, and a restart script. The component in the Oracle tier is a database script.

Example of a two tier application

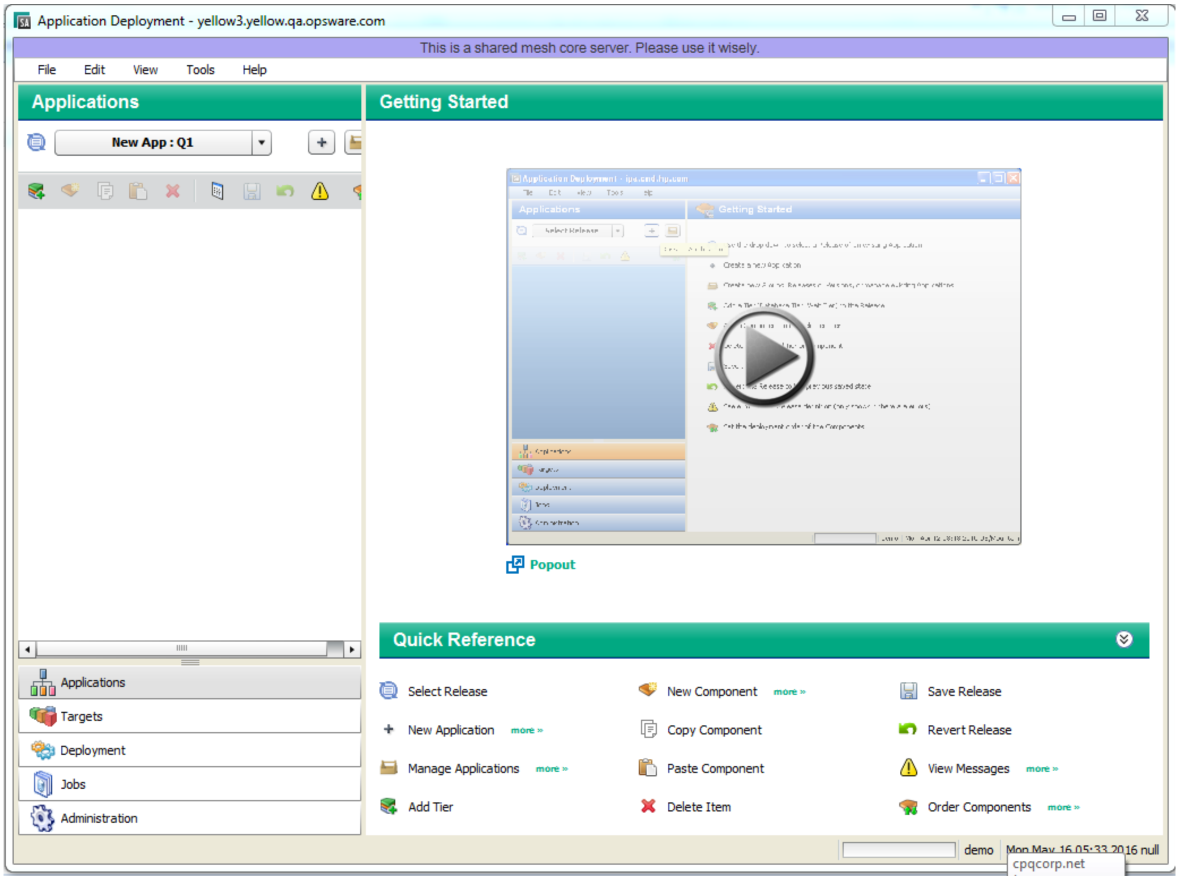


The following instructions assume that you are creating a new application. If you want to work with an existing application, see "Applications overview" on page 454.

Before you can create an application, Your SA administrator must grant you permission to access the Application Deployment tool and to create applications. See "Setting permissions" on page 579.

To start the Application Deployment tool:

1. Start the SA Client. For more information, see the SA 10.50 User Guide.
2. In the SA Client, select the **Tools > Application Deployment** menu item. The Application Deployment interface opens with the Applications screen visible, as shown here:



This is a shared mesh core server. Please use it wisely.

Click the ▶ button to start a short video demo that demonstrates the function of the Applications screen. To play the video in a separate window, click Popout.

The Quick Reference area lists the functions of the tool bar buttons available on this screen. Click the more>> links to open related online help topics in a separate browser window.

To create and configure a new application:

1. Go to the Applications screen (click **Applications** in the lower left corner).
2. Click **Create a New Application** ⊞. The Create New Application dialog opens. Note that the ability to create applications is a specific action privilege in SA that needs to be assigned to a group to which you belong (see "Setting permissions" on page 579).
   a. Enter the name of your new application.
   b. *Optional*: Assign your new application to an existing application group.
   c. *Optional*: Enter a description of your new application.
   d. In the **Release Name** field, enter the name of the current release that you are working on for this application.

      For example, if you release quarterly, you could use Q3-2010, Q4-2010, Q1-2011, Q2-2011, and so forth as your release names.
   e. Select a life cycle for this release. The life cycle specifies which phases your application goes through on its way from development to production.

      For example, a common life cycle is Development > QA > Pre-Production > Production. This is the default Standard life cycle.
   f. Click **OK** to save the application information.
3. Click **Add Tier** to add one or more tiers to your application.
   a. Select the type of tier to add.

      A group of standard tiers are provided with your software. If a tier that you need is not available, your SA administrator may need to add it. See the SA 10.50 Administration Guide"Administering Application Deployment" on page 593.
   b. Click **Add**.
   c. Repeat steps (a) and (b) for each additional tier that you need.
4. To add components to a tier, follow these steps:
   a. Select the tier.
   b. Click the “New Component” button to add components to this tier.
   c. Enter a name for the new component.

   d. Select the component type. See "Components" on page 468 for more information.

   e. Click **OK**.

   f. Enter the information required to fully specify the location and behavior of the new component.

      The information that you must provide varies depending on the type of component that you are adding. Some component types require instructions for rollback and undeploy scenarios. See "Types of components" on page 470 for more information.

   g. Repeat steps (b) through (f) above to add more components to this tier.

   h. Click **Save Release** 💾 to save your changes.

5. Repeat steps 3–4 above to add more tiers and add more components to each tier.

6. Click **Edit component deployment order** .

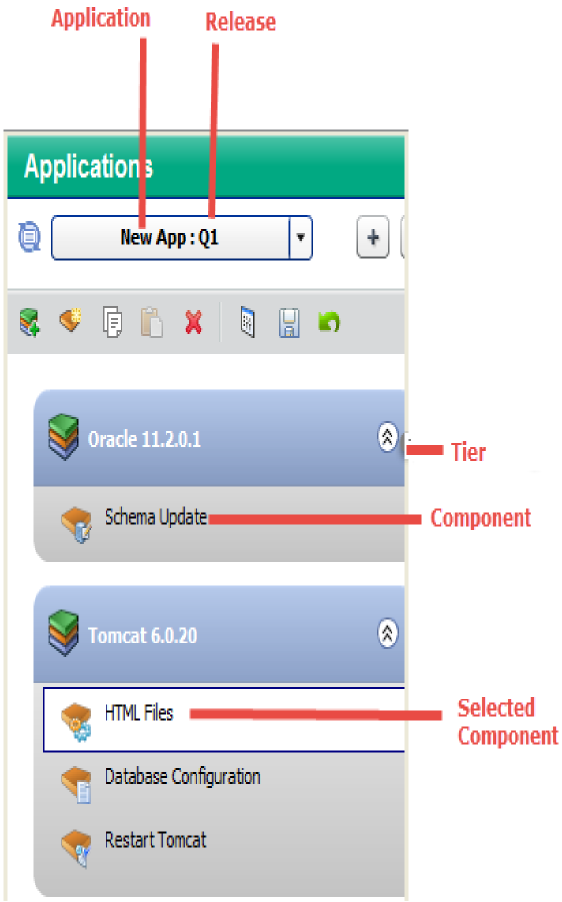
7. Use the green arrow keys to arrange your components in the proper deployment order. See "Changing the deployment order of components" on page 495 for details.

After you have added all tiers and components, your application is fully defined. Your application should look similar to the example shown in the following figure.

If the “Warning” sign ⚠ appears on the tool bar, this indicates that a component is not fully defined or is incorrectly defined. For example, a Code component’s source location might reference a directory that does not exist.

Click the ⚠ to determine what information or action is required.

Example of an application with three tiers and seven components

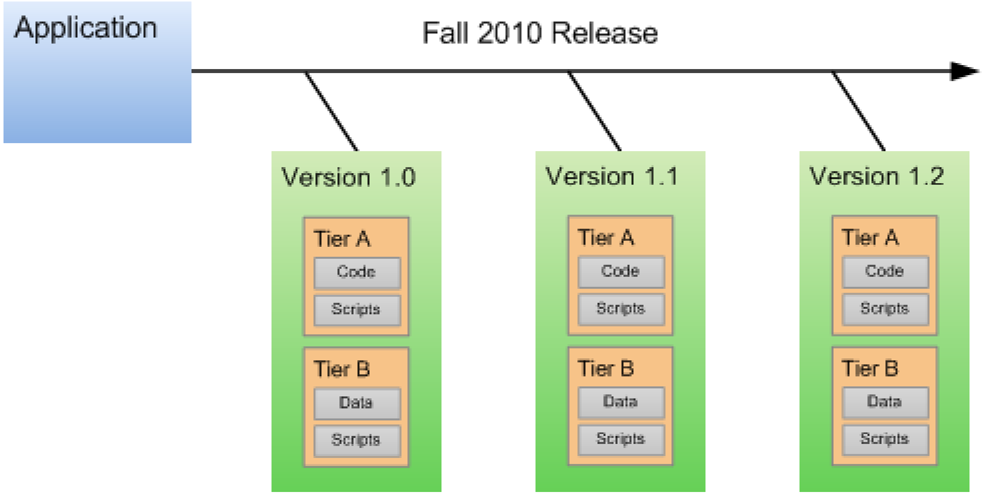


Step 2: Create a version to deploy

After you have defined your application, you must create a new **version** of the application. You then deploy the components that constitute that version to your targets.

The figure below shows an application that is being developed for a release named “Fall 2010.” Three versions of this application have been built: versions 1.0, 1.1 and 1.2. Any of these three versions can be deployed to a target.

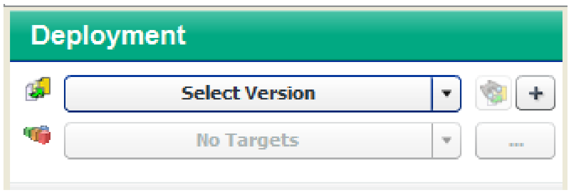
Applications, releases, and versions



Make sure that you have successfully created a version of your application before you attempt to deploy it.

To create a new version of your application:

1. Go to the Deployment screen (click **Deployment** in the lower left corner).
2. In the Select Version drop-down field, navigate to your application.



3. Under the pertinent release of your application, click the **Create New Version** link.
4. In the Create New Version window, enter a Version name (or number).

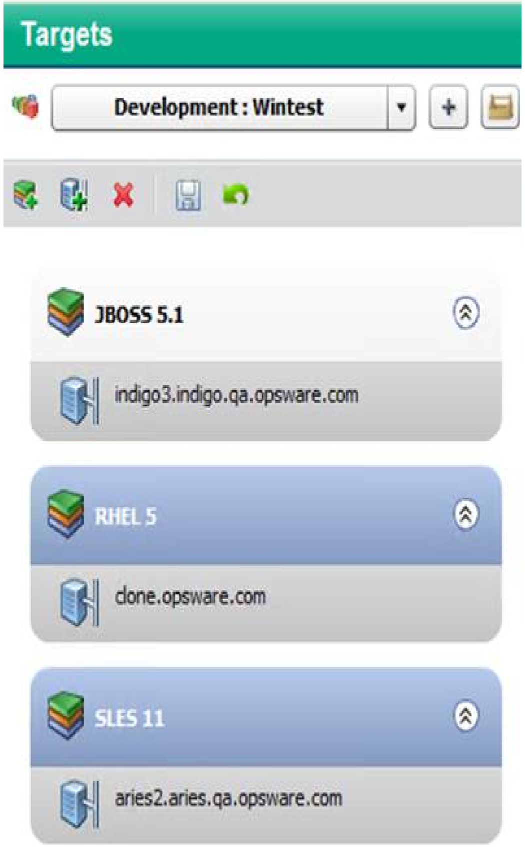
   The version name is simply a string that you specify. If the previous version of this release contains a number, Application Deployment will automatically increment it. If the previous version was “1,” for example, “2” will appear in the Version field. If the previous version was “V 1.2.1,” the string “V 1.2.2” will appear in the Version field.
5. Regardless of what Application Deployment puts in the Version field, you can specify any version name that you want.
6. *Optional:* In the Description field, enter any additional information that you want to specify for this version.

7. *Optional:* If you want to see a list of the files included in the Code components, select **Show Code Component Changes**.
8. Click **Create**. This creates a new version of your application by gathering all the artifacts that are components of this application.
9. Click **Close**. The Application Deployment tool has created a new version of your application that is ready for deployment.

Step 3: Define your targets

Targets are sets of servers to which your application will be deployed. Each target must include the **tiers** that your application requires.

Target with three tiers



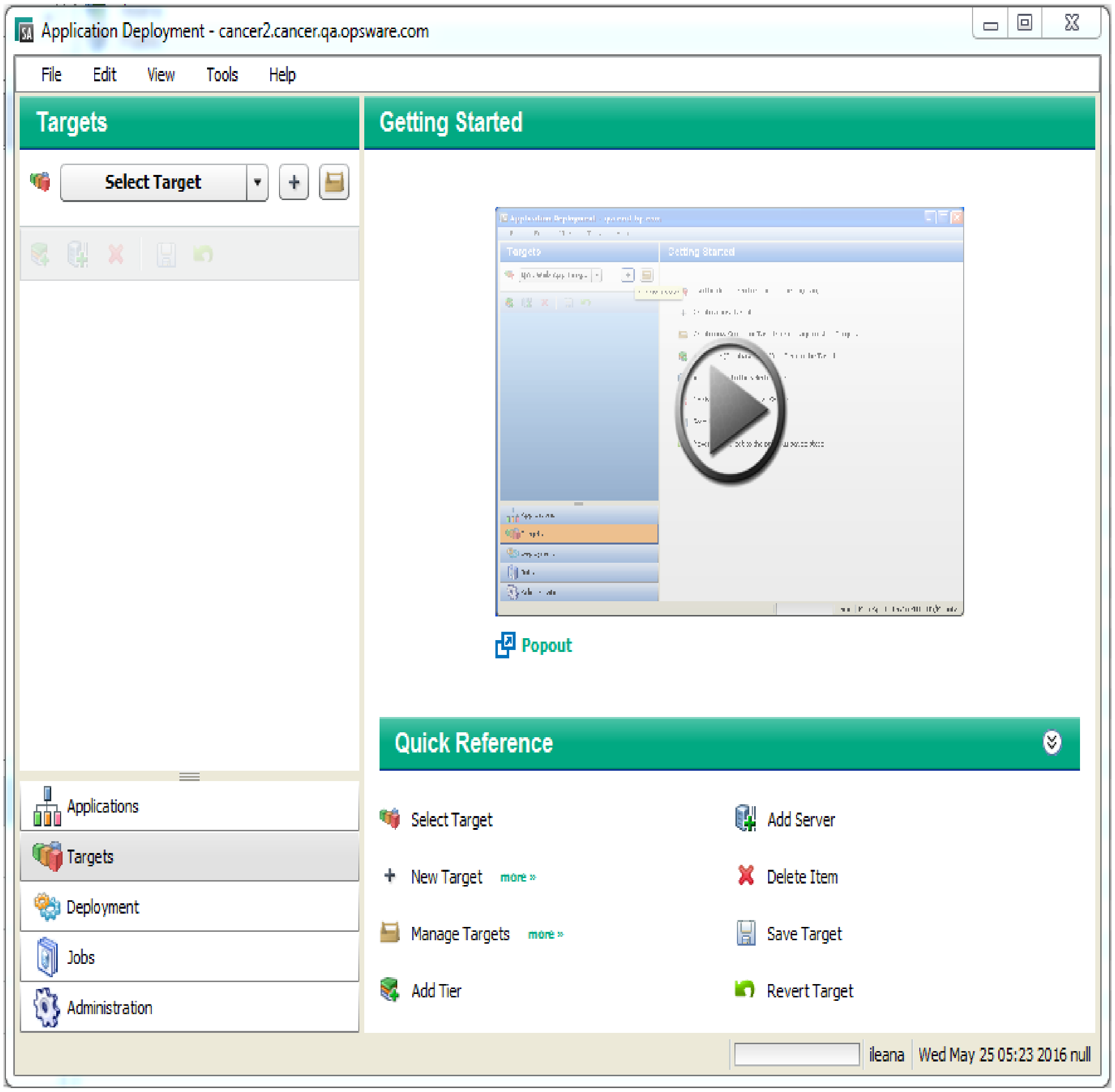
Two levels of permission are required before you can successfully set up targets. Your SA administrator must grant you Write permission to the pertinent environment (see "Environment permissions" on page 591).

Before you can add servers to tiers, you must also have permission to work with those servers. This is managed in the SA client (see "Action permissions" on page 584).

Targets are defined on the Targets screen in the Application Deployment tool.

When you are accessing the Target screen for the first time, it will be displayed as shown in the following figure:

Targets Screen



Click the ▶ button to start a short video demo that demonstrates the function of the Targets screen. To play the video in a separate window, click Popout.

The Quick Reference area lists the functions of the tool bar buttons available on this screen. Click the more>> links to open related online help topics in a separate browser window.

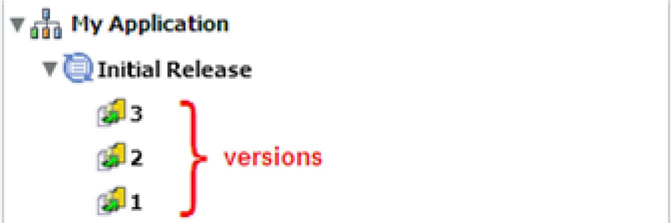
The following instructions assume that you are creating a new target. These instructions also assume that, for the purpose of this Quick Start, your target will include only servers.

To create and configure a new target:

1. Go to the Targets screen (click **Targets** in the lower left corner).
2. Click the “Create a New Target” button: + . The Create New Target dialog opens.
3. Enter the target name. Target names must be unique within the environment.
4. In the Location drop-down field, select the environment that will contain this target.

   There are two ways to populate your target with tiers: you can add them manually (as described in step 6), or you can create them automatically.

   To automatically create the same set of tiers in your target that are used in an existing application, select “Use Application's Tiers” and select a version of the application.

   > **Note:** A specific version of a particular release of the application must have been previously created before it will appear in the Select Version drop-down list.

5. Click **OK**.
6. Follow these steps to add each tier that your application requires:
   a. Click the **Add Tier** button.

   b. Select the type of tier to add. You must create the same set (or a superset) of tiers that are used by your application.

      A group of standard tiers are provided with your software. If a tier that you need is not available, your Application Deployment administrator may need to add it.

c. Click the **Add** button.

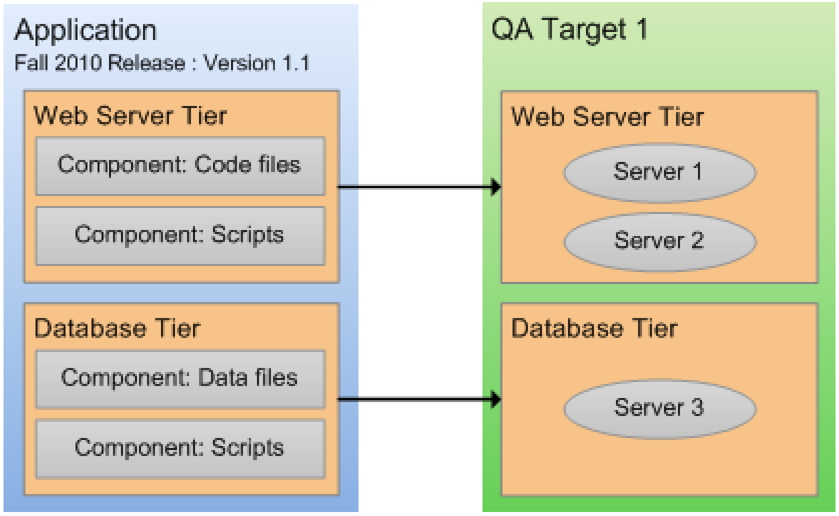
7. For each tier that you added in step 6, follow these steps to specify the server (or servers), that will run this tier:
   a. Select the tier that you want to work with.
   b. Select the **Add Server** button to add a server to this tier.

      The list of available servers is filtered by platform. Make sure that you select suitable servers for your tier. For example: You should select only Red Hat Enterprise Linux 5 servers if your components for that tier are intended for Red Hat Enterprise Linux 5.
   c. In the Add Servers to Target dialog, locate and select the managed server that you want to add. If you are setting up a multi-server tier, use the CTRL key to select multiple servers.
   d. Click **OK**.
8. Repeat step 7 to add servers to all remaining tiers.
9. Click the **Save Target** button to save your changes.

Step 4: Deploy a version to a target

You deploy a **version** of your application to a **target**. This copies the application files to the target servers and executes any scripts that you provided with your application.

Deploying an application to a target



After you create a version, you can deploy that version to a target.

To deploy a version:

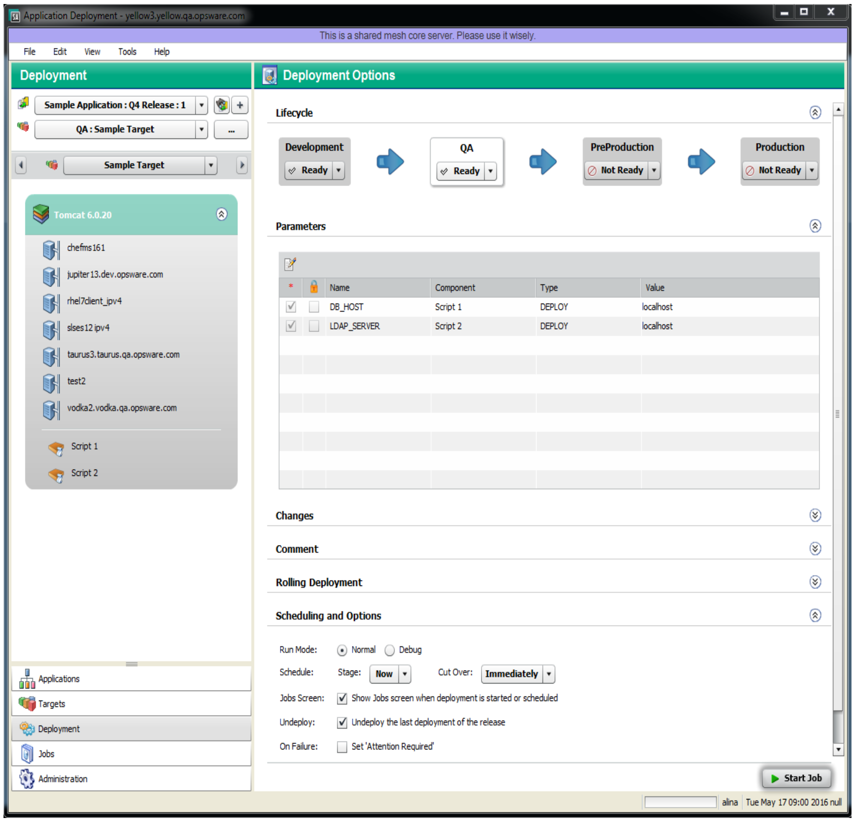
1. Go to the Deployment screen (click **Deployment** in the lower left corner).
2. From the Select Version drop-down list, select the version that you created—if you just created this version, it will already be selected.
3. From the Select Target drop-down list, select the target that you created earlier. This specifies the target server (or servers) where you want the application to be deployed.

   If you do not see your target in the list, make sure that the following conditions are true:
   - The tiers of the target match (or are a superset of) the application tiers.
   - You have Deploy permission for this environment.
   - Each tier contains at least one server.
   - The life cycle for this release includes the environment where your target resides.
   - The environment is *not* marked Not Ready in the life cycle.

For additional target options, click the ... button to the right of the Select Target drop-down list. See "Deploying an application" on page 533 for more information about these options.

After you select a version and a target, the Deployment screen looks like this—the left panel presents a preview of what will be deployed, and the right panel shows you the deployment job settings.

4. *Optional:* Modify the deployment job settings in the right panel. See "Deploying an application" on page 533 for details.



*	🔒	Name	Component	Type	Value
☑	□	DB_HOST	Script 1	DEPLOY	localhost
☑	□	LDAP_SERVER	Script 2	DEPLOY	localhost

Run Mode: ● Normal ○ Debug

Schedule: Stage: Now Cut Over: Immediately

Jobs Screen: ☑ Show Jobs screen when deployment is started or scheduled

Undeploy: ☑ Undeploy the last deployment of the release

On Failure: □ Set 'Attention Required'

5. Click **Start Job**. This launches a Deployment job that will copy your application components to the target server and run any necessary code and scripts.

   You can see the status of the deployment on the Jobs screen.

Applications overview

Applications are one of the basic building blocks in Application Deployment. They have two primary qualities: their **content** and their **structure**. The content of an application determines its function. The structure of an application reflects its organization. In the context of Application Deployment, the structure helps associate the application with suitable, similarly structured application deployment targets.

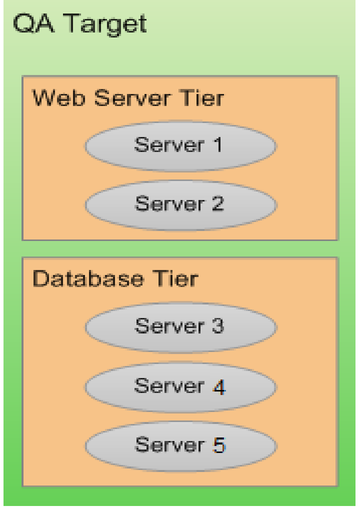
The structure also specifies the order in which the **components** are deployed. In Application Deployment, components implement the content of an application. The following types of components are available:

Code components (for example: HTML files, JAR files, WAR files, .NET assemblies, or databases)

- Ad hoc scripts
- Configuration files
- Application Configurations
- Software Policies
- Packages
- HPE Operations Orchestration (OO) flows
- Registry components (for Windows targets only)

Tiers are containers that represent the structure of an application. A tier includes the components that a particular part of the application requires. Examples of tiers are web servers, application servers, and databases. **Targets** are also organized in tiers. When you deploy an application, the components of each application tier are deployed to the target servers in the corresponding tier.

Application with two tiers



A single application can have multiple **releases**. A release, in turn, typically has multiple **versions**. A version is an immutable “snapshot” of a release at a specific point in time. When you deploy an application to an environment, you deploy a specific version.

There are two types of releases, **full** releases and **delta** releases, which differ in their treatment of Code components. For a full release, all files belonging to all Code components are included. For a delta release, only those files that have been added or modified since a specific version of the release was created are included. Files that have been deleted will be deleted on the target servers during deployment.

When you specify the components included in a release, you can specify both **rollback** and **undeploy** instructions for each component. The rollback instructions are used in the event that a failure occurs during deployment. In this situation, Application Deployment will retrace its steps and undo each step of the deployment according to the rollback instructions that you provide. The undeploy instructions are used to uninstall an application.

The degree to which rollback and undeploy are automated depends on the type of component. Code components, for example, use standard scripts that are maintained by the Application Deployment administrator. Script, and OO Flow components, in contrast, require you to provide explicit rollback and undeploy instructions.

You can create application **groups** to help you organize your applications in the most convenient manner. This is an organizational tool that is provided strictly for your benefit; it does not affect the deployment of applications.

The process of defining a new application has three primary steps:

- Create the application and the release
- Specify the tiers that are included in the release
- Specify the components for each tier

Prerequisites

Before you can define an application, the following conditions must be met:

The tiers that this application requires are available in the Application Deployment user interface.

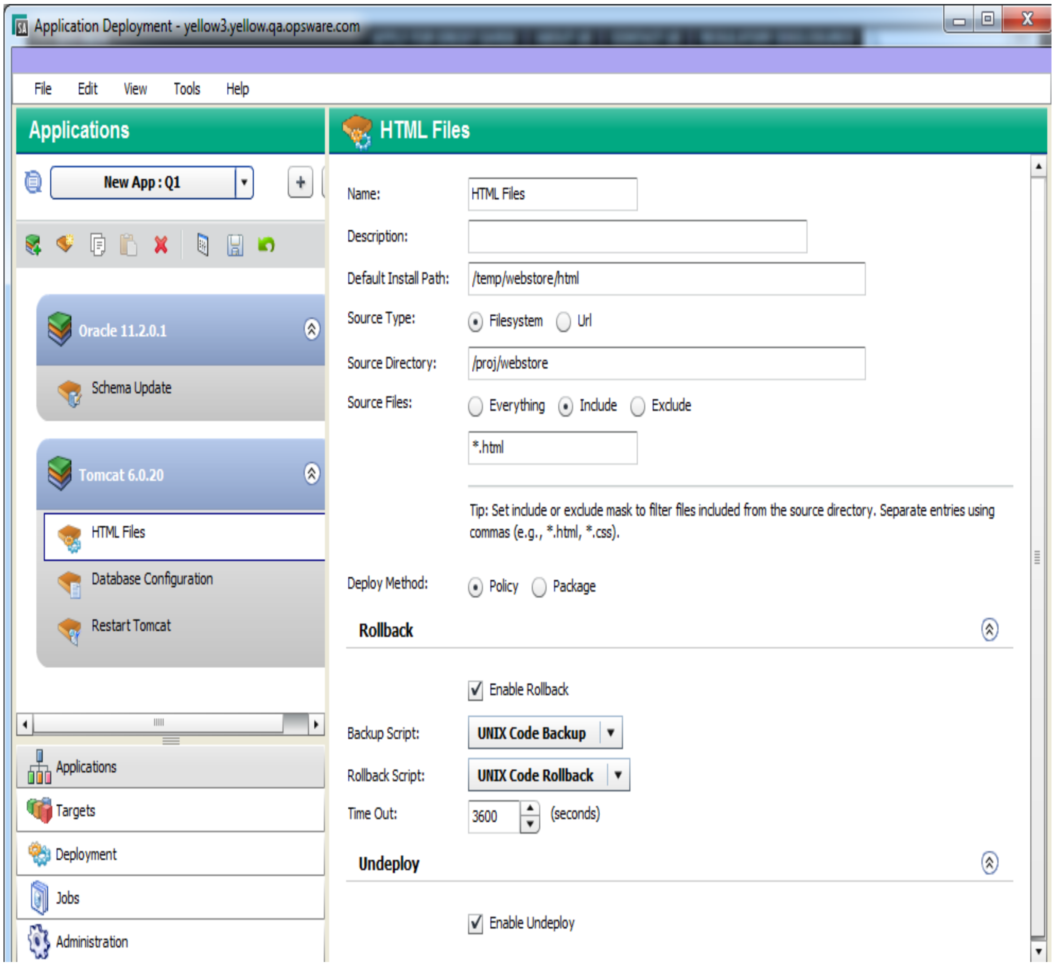
You have permission to create applications.

See the SA 10.50 Administration Guide for more information.

About the Applications screen

After an application has been defined, the Applications screen looks like this:

Applications tab with existing applications



Name: HTML Files

Description:

Default Install Path: /temp/webstore/html

Source Type: ● Filesystem ○ Url

Source Directory: /proj/webstore

Source Files: ○ Everything ● Include ○ Exclude

*.html

Tip: Set include or exclude mask to filter files included from the source directory. Separate entries using commas (e.g., *.html, *.css).

Deploy Method: ● Policy ○ Package

Rollback

☑ Enable Rollback

Backup Script: UNIX Code Backup

Rollback Script: UNIX Code Rollback

Time Out: 3600 (seconds)

Undeploy

☑ Enable Undeploy

Things to note:

- Application names must be unique.
- All defined applications and releases that you have permission to view are listed in the drop-down list in the upper left corner. Here, the New App: Q1 was selected.
- The left pane shows you the structure of the application.
- The right pane displays information about the item selected in the left pane. You can select a single component or a whole tier. When you select an item, the background color changes for that item. Here, the HTML Files component is selected.
- You can drag and drop components between tiers (assuming the platform is the same).
- You can change the deployment order of the components (see "Changing the deployment order of components" on page 495).

About tiers

Tiers are containers that are used to represent the **structure** of an application. Tiers contain components, which implement the functions of the application. The Application Deployment administrator defines the tiers that are available for you to use (see the SA 10.50 Administration Guide). A collection of commonly used tiers are provided “out of the box.” Your administrator will probably need to modify them to suit your environment and add new tiers, as well.

To view a simple example, see "Quick start" on page 440.

Working with applications

Creating a new application

This topic provides basic instructions for creating a new application. It assumes that you have the HPE Server Automationpermissions required to create applications. See "Setting permissions" on page 579 for details.

To create and configure a new application:

1. Go to the Applications screen (click **Applications** in the lower left corner).
2. Click the **Create a New Application**. The Create New Application dialog opens. Note that the ability to create applications is a specific action privilege in SA that needs to be assigned to a group to which you belong (see "Setting permissions" on page 579).
   a. Enter the name of your new application (see"Naming rules" on page 438).
   b. *Optional*: Assign your new application to an existing application group.
   c. *Optional*: Enter a description of your new application.
   d. In the **Release Name** field, enter the name of the current release that you are working on for this application.

      For example, if you release quarterly, you could use Q3-2010, Q4-2010, Q1-2011, Q2-2011, and so forth as your release names.

   e. Select a lifecycle for this release. The lifecycle specifies which phases your application goes through on its way from development to production.

      For example, a common lifecycle is Development > QA > Pre-Production > Production. This is the default Standard lifecycle.

   f. Click **OK** to save the application information.

3. Click the **Add Tier** to add one or more tiers to your application.

   a. Select the type of tier to add.

      A group of standard tiers are provided with your software. If a tier that you need is not available, your Application Deployment administrator may need to add it. See the SA 10.50 Administration Guide.

   b. Click **Add**.

   c. Repeat steps (a) and (b) for each additional tier that you need.

4. To add components to a tier, follow these steps:

   a. Select the tier.

   b. Click **New Component** to add components to this tier.

   c. Enter a name for the new component.

   d. Select the component type. See "Types of components" on page 470 for more information.

   e. Click **OK**.

   f. Enter the information required to fully specify the location and behavior of the new component.

      The information that you must provide varies depending on the type of component that you are adding. Some component types require instructions for rollback and undeploy scenarios.

   g. Repeat steps (b) through (f) above to add more components to this tier.

   h. Click **Save Release** to save your changes.

5. Repeat steps 3–4 above to add more tiers and add more components to each tier.

6. Click **Edit component deployment order**.

7. Use the green arrow keys to arrange your components in the proper deployment order (see "Changing the deployment order of components" on page 495).

After you have added all tiers and components, your application is fully defined. Your application should look similar to the example shown in "Quick start" on page 440.

If the “Warning” sign ⚠ appears on the tool bar, this indicates that a component is not fully defined or is incorrectly defined. For example, a Code component’s source location might reference a directory that does not exist.

Click the ⚠ icon to determine what information is needed.

Managing applications

If you have permission to do so (see "Overview" on page 579), you can use the Manage Applications tool to perform certain tasks. This tool enables you to create versions that can be deployed to targets. It also enables you to specify very finely grained information about applications and releases.

To open the Manage Applications tool, click the button on the Applications screen. If you do not have permission to manage applications (see "Setting permissions" on page 579), this button is not available.

The Manage Applications tool contains a table that lists the items shown here:

Application groups (can be nested)

Applications

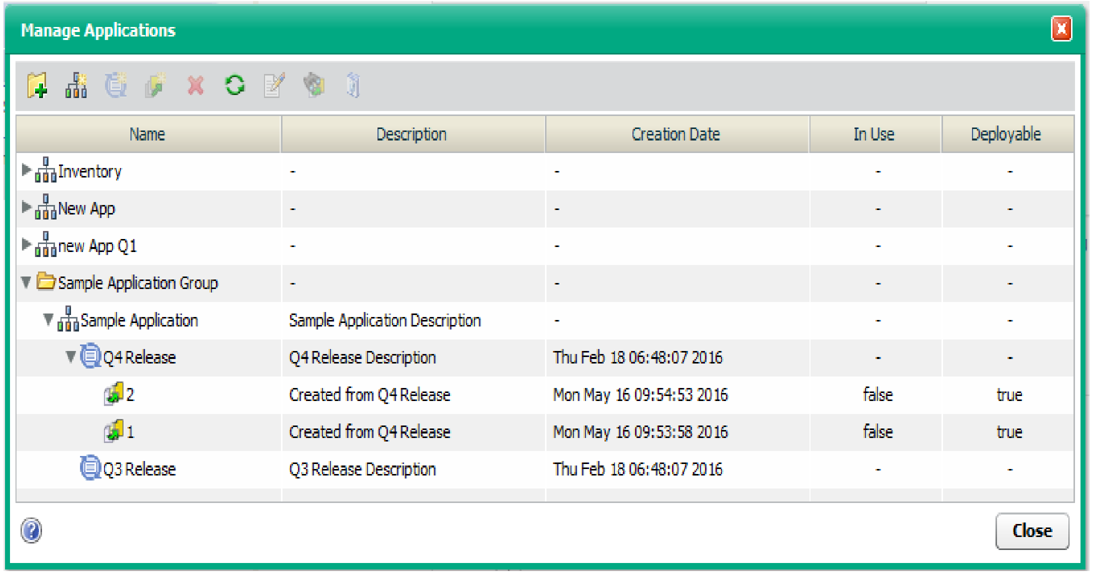
Releases

Versions

Only those applications that you have permission to View are visible.

The Name and Description columns show the information that was specified when the item was created (or most recently edited). The Creation Date is only shown for releases and versions. The In Use column indicates whether a version has been deployed (and not undeployed or rolled back). The Deployable column indicates whether or not this version was successfully created.

Manage Applications tool



Name	Description	Creation Date	In Use	Deployable
Inventory	-	-	-	-
New App	-	-	-	-
new App Q1	-	-	-	-
Sample Application Group	-	-	-	-
Sample Application	Sample Application Description	-	-	-
Q4 Release	Q4 Release Description	Thu Feb 18 06:48:07 2016	-	-
2	Created from Q4 Release	Mon May 16 09:54:53 2016	false	true
1	Created from Q4 Release	Mon May 16 09:53:58 2016	false	true
Q3 Release	Q3 Release Description	Thu Feb 18 06:48:07 2016	-	-

To edit a specific item, select the row containing that item, and click the button on the tool bar—or simply double-click the item.

Because a version is an immutable “snapshot” of a release at a particular point in time, versions cannot be edited.

Creating a version of an existing release

Before you can deploy a release, you must create a version of that release. A version is a “snapshot” of the release and the lifecycle that is associated with that release at the time that the version is created. The release may continue to evolve, but the version will not change. This ensures that a specific version of a release can be repeatedly deployed.

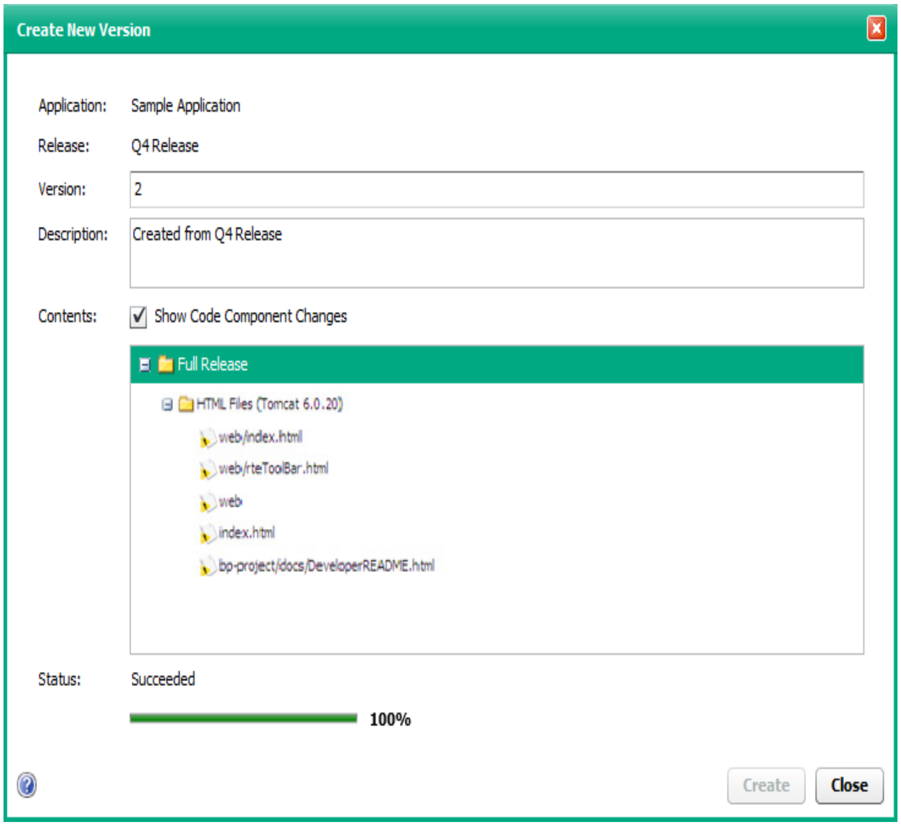
If you have the proper permissions (see "Overview" on page 579), you can create a version of a release.

Note that versions can also be created on the Deployment screen. See "Deploying an application" on page 533 for more information.

To create a version of an existing release:

1. Open the Manage Applications tool (see "Managing applications" on the previous page).
2. In the table, select the row containing the release that you want to work with.
3. Click the “Create Version” button. The Create New Version dialog opens.

4. Specify the following information:
   a. The Version shown is based on the name (which is frequently a number) of the previous version. You can change this if you like.
   b. *Optional*: The Description initially indicates which release the version is based on. You can also change this if you like.
   c. *Optional*: Select the Show Code Component Changes box if you want to see a list of the Code component files. For a full release, all files are listed. For a delta release, only those files that have changed since the “baseline” version are listed. For example:



Application: Sample Application

Release: Q4 Release

Version: 2

Description: Created from Q4 Release

Contents: ☑ Show Code Component Changes

```
Full Release
    HTML Files (Tomcat 6.0.20)
        web/index.html
        web/rteToolBar.html
        web
        index.html
        bp-project/docs/DeveloperREADME.html
```

Status: Succeeded

100%

5. Click **Create**. Application Deployment creates the version.
6. When the status displays “Succeeded,” click **Close**.
7. If the job fails, click **Close**, and go to the Jobs screen to determine why.

Creating a new application group

An applications group is a container that you can use to organize your applications in a hierarchy. If you have the proper permissions (see "Overview" on page 579), you can create applications groups.

To create an applications group:

1. Open the Manage Applications tool (see "Managing applications" on page 460).
2. To create an application group at the top level, do not select anything. To create a nested application group, select the group under which you want to create the new group.
3. Click **Create Application Group**.
4. Specify a name for your new group (see "Naming rules" on page 438). This name must be unique within this level of the application hierarchy.
5. Click **OK**. Your new group should now be visible in the table. The groups and applications are listed in alphabetical order at each level of the hierarchy.

You can drag and drop applications between groups in the Manage Applications tool.

Creating a new application

If you have the proper permissions (see "Overview" on page 579), you can create a new application by using the Manage Applications tool. This is an alternative to creating an application on the main Applications screen. This is useful, for example, if you want to set the permissions for the application before you add tiers and components.

To create a new applications in the Manage Applications tool:

1. Open the Manage Applications tool (see "Managing applications" on page 460).
2. To create an application at the top level, do not select anything. To create the application within an existing application group, select the group that will contain the new application.
3. Click **Create Application**.
4. Specify the following information:
   a. Provide a unique name for your new application (see "Naming rules" on page 438).
   b. *Optional*: Enter a description of your new application.

   c. In the Release Name field, enter the name of the current release that you are working on for this application. By default, this will be "Initial Release."

   d. Select a lifecycle for this release. The lifecycle specifies which phases your application goes through on its way from development to production.

      For example, a common lifecycle is Development ® QA ® Pre-Production ® Production. This is the default Standard lifecycle.

5. Click **OK**. Your new application should now be visible in the table. The applications and groups are listed in alphabetical order at each level of the hierarchy.

Next steps:

If you want to specify permissions for this application, keep the Manage Applications tool open, and see "Setting permissions" on page 579.

If you want to specify tiers and components for the application, close the Manage Applications tool, and select the release that you just created on the main Applications screen.

Creating a new release of an existing application

If you have the proper permissions (see "Overview" on page 579), you can use the Manage Applications tool to create a new release for an existing application. The new release is cloned from an existing release. It can be either a full release or a delta release. A full release contains all of the files included in all of the Code components; a delta release contains only those Code component files that have been added or modified since the specified "baseline" version was created.

For more information, see "Managing applications" on page 460.

To create a new release for an existing application:

1. Open the Manage Applications tool (see "Managing applications" on page 460).
2. In the table, select the row containing the application that you want to work with.
3. Click the "Create Release" button: . The Create New Release dialog opens.
4. Specify the following information:
   a. From the Clone Release drop-down list, select the release that you want to clone. The new application will contain the same tiers and components as the release that you select.

   b. Enter a Name for this release (see "Naming rules" on page 438). The Name must be unique within this application.

   c. *Optional*: Provide a Description for the release. This information appears on reports and is also displayed in the Manage Applications tool.

   d. *Optional*: Specify a Change Request number for this release. This information also appears in reports.

   e. *Optional*: In the ROI box, estimate the “return on investment” that you expect to realize for each target machine. This is your own assessment of how much value you will derive by deploying this release. You define the scale and meaning of this field. It is used in the HPE Server AutomationROI reports. You can specify any positive whole number.

   f. *Optional*: Specify the Planned Release Date. This information also appears in reports.

   g. If this is a delta release, select the Delta check-box, and select the “baseline” version from the drop-down list. Only those files located in the Source Directory specified for each Code component that have been added or modified since the baseline version was created will be deployed.

      If this is a full release, be sure that the Delta check-box is not selected.

   h. Select the pertinent Lifecycle from the drop-down list.

5. Click **OK**.

Modifying properties of an existing release

If you have the proper permissions (see "Overview" on page 579), you can modify the properties of any of its releases.

To modify the properties of an existing release:

1. Open the Manage Applications tool (see "Managing applications" on page 460).
2. In the table, select the row containing the release that you want to modify.
3. Click the “Edit Properties” button: 📝.
4. Modify the properties of the release. For descriptions of the properties, see "Applications overview" on page 454.
5. Click **OK**.

Deleting application groups, applications, releases, or versions

If you have the proper permissions (see "Overview" on page 579), you can use the Manage Applications tool to delete applications, groups, releases, or versions.

To delete an existing application, group, release, or version:

1. Open the Manage Applications tool (see "Managing applications" on page 460).
2. In the table, select the row containing the items that you want to delete. Note the following:
   - You can select multiple items by using the Ctrl or Shift key. You can only delete multiple items of the same type, however—for example, multiple applications or multiple releases.
   - If you delete an application, all releases and versions of that application will also be deleted, as will any jobs that are scheduled but not running
   - You cannot delete a version that is In Use.

     You must first “undeploy” this version from the Jobs screen (see "Managing Application Deployment jobs" on page 555). Then you can delete it.

     Click **Show Deployment Jobs** to see the list of jobs that must be undeployed or rolled back before you can delete this version.

     Note that, unless you are the Application Deployment administrator, you can only undeploy or roll back deployments that you initiated.
   - Some application groups cannot be deleted.
3. Click **Delete**.

There is no UNDO. You will be asked if you really want to delete the item (or items). Click **Yes** to delete them.

Renaming an application or application group

If you have the proper permissions (see "Overview" on page 579), you can change the name of an application or application group by using the Manage Applications tool.

To rename an application or group:

1. Open the Manage Applications tool (see "Managing applications" on page 460).
2. In the table, select the row containing the application or group that you want to rename.
3. Click **Edit Properties**.
4. Enter a new unique name for the application or group (see "Naming rules" on page 438).
5. Click **OK**.

Granting or revoking permissions for an application

When you create an application, you can specify which SA users and user groups are allowed to view, edit, and deploy your application. If you do not explicitly specify these permissions, you and your Application Deployment administrator will be the only users who can access your application. See "Overview" on page 579 for more information.

To set the permissions for an application:

1. Open the Manage Applications tool (see "Managing applications" on page 460).
2. In the table, select the row containing the application that you want to work with.
3. Click the **Edit Properties**.
4. From the Choose User menu, select a user or user group.
5. Click **Add User**.
6. Select the permissions that you want to grant:
   a. View permission allows the user to see your application in the Select Release drop-down list on the Applications screen.
   b. Edit permission allows the user to change the content (components), structure (tiers), or properties (name, description, release name, etc.) of your application.
   c. Deploy permission allows the user to deploy your application to one or more targets.

   You can select or deny any combination of these permissions. View is automatically selected when you select Edit or Deploy.
7. To remove a user (or group) from the permissions list, select the row containing that user (or group), and click **Delete** .
8. Click **OK** to close the Edit Applications dialog and save your changes. The **Cancel** button discards your changes.

Viewing the properties of a specific version

If you have the proper permissions (see "Overview" on page 579), you can open the Version Viewer from the Manage Applications tool (see "Viewing the properties of a version" on page 548).

To open the version viewer from the Manage Applications tool:

1. Open the Manage Applications tool (see "Managing applications" on page 460).
2. In the table, select version that you want to view.
3. Click **View version** on the tool bar. The Version Viewer opens.

Viewing the deployment jobs that use a specific version

If you have the proper permissions (see "Overview" on page 579), you view a list of the deployment jobs that use the most recent version of a release.

To view the list of deployment jobs that use a specific version:

1. Open the Manage Applications tool (see "Managing applications" on page 460).
2. In the table, select version that you want to view.
3. Click the **Show Deployment Jobs** on the tool bar. The Jobs screen opens and displays a list of all deployment jobs that use this version.

Components

Overview

You can create and modify components in applications that you have permissions to edit (see "Setting permissions" on page 579).

To view an example of an application with simple components, see "Quick start" on page 440.

About components

Components represent the **content** of an application. The components implement the various functions of the application. A component is associated with a specific tier. There are eight types of components that you can use to populate the tiers of an application.

Component type	Icon	Purpose
Code		**Copies files** to a target server. For example, this might be a set of HTML files that implement a web-based store. See "Code components" on the next page.
Script		Runs a script on a target server **during the deployment** of an application. For example, you might have two scripts: one that shuts down all applications before the deployment, and another that starts the applications up again after deployment. See "Script components" on page 474.
Configuration File		Creates a new configuration file on a target server. These files typically contain **very specific information** for a particular application. For example, you might need to create a configuration file that specifies information about a database. See "Configuration File components" on page 476.
Application Configuration		Deploys existing **Application Configurations** from the SA Library to a target server. See "Application Configuration components" on page 479.
Software Policy		Attaches a **Software Policy** from the SA Library to a target server. Alternatively, you can specify that the policy be installed instead of attached. See "Software Policy components" on page 483.
Package		Installs a **Software Package** from the SA Library on a target server. Alternatively, you can specify that the package be attached as a policy. See "Package components" on page 484.
OO Flow		Initiates an **HPE Operations Orchestration (OO)** workflow. For example, the workflow might be that whenever a specific application is deployed, an email is sent to the product owner. See "OO flow components" on page 488.
DMA Flow		Initiates an **HPE Database and Middleware Automation (DMA)** workflow on a target server, instance, or database. For example, you could use a DMA workflow to install critical database patches.
Windows Registry		Adds or deletes **Windows registry** keys or values. This type of component is only available for Windows tiers. See "Windows Registry components" on page 490.

Components are executed against servers. By default, components are deployed in the order in which they were created. You can modify this order if you have permission to edit the application (see "Changing the deployment order of components" on page 495).

Types of components

Following are the different types of Application Deployment components:

- "Code components" below
- "Script components" on page 474
- "Configuration File components" on page 476
- "Application Configuration components" on page 479
- "Software Policy components" on page 483
- "Package components" on page 484
- "OO flow components" on page 488
- "Windows Registry components" on page 490

OO Flow components are only available if you have configured your SA core server to integrate with an HPE OO server.

Windows Registry components can only be used in Windows tiers.

Code components

Code components are used to deploy files to managed servers. A Code component is simply a package of files that Application Deployment creates from a specified repository. This repository can be any of the following:

- The file system of the SA core server.
- A single file accessed via a URL
- A source code control system

Code components are dynamic. They capture the current contents of a directory (or URL) each time that a version is created. In contrast, Software Policy or Package components use the same, relatively static item from the SA Library each time.

The maximum number of files that a Code component can contain is 100,000.

When you create a Code component, specify the following information:

Code component properties

Property	Required?	Purpose
Name	Yes	Name of the component. This name appears in the tier. It must be unique within the application.
Description	No	Description that appears in reports.
Base Path	No	The Base Path is combined with the Default Install Path to form the Full Install Path—where the files will be placed on the managed server. The Base Path and Full Install Path are only visible when Sandboxing is enabled (see Administering Application Settings). See the note immediately following this table.
Default Install Path	Yes	Directory where the files will be placed on the target (within the Base Path if Sandboxing is enabled).
Full Install Path	Read-only	Base Path + Install Path (only visible when Sandboxing is enabled).
Source Type	Yes	Where the source files are located: Filesystem – Look for the files in the file system of the SA core server. This includes any Samba and NFS mount points. CVS or Subversion – Look for the files in the specified source code control system (see "Using a source code control System" on page 473). URL – Look for a single file according to the protocol specified in the URL (see "Using a URL" on page 474). These locations are maintained by your Application Deployment administrator. See "Administering code component source types" on page 603 for more information.
Source Directory	Yes	Directory where the source files are located in the repository. This directory must be mounted on the SA core server. If you specify URL for the source type, there is no Source Directory. There is a URL property instead.
Source Files	Yes	Everything – Get all files in the Source Directory. Include – Get all files in the Source Directory whose names match the pattern that you specify. Exclude – Get all files in the Source Directory *except* those

Code component properties, continued

Property	Required?	Purpose
		that match the pattern that you specify. See "Using filters" on the next page.
Deploy Method	Yes	When you create a new version, Application Deployment creates a package containing the files specified and any scripts required. Policy – Application Deployment also creates a policy when it creates a new version. When the version is deployed, it will attach and remediate this policy on each managed server included in the target. Package – When the version is deployed, Application Deployment will perform an “ad hoc” install of the package on each managed server included in the target.
Backup	No	The Backup and Rollback scripts available depend on the platform of the tier. These scripts are maintained by your Application Deployment administrator. See "Administering scripts" on page 601 for more information.
Rollback	No	
Time Out	Yes	Specifies the maximum time (in seconds) that SA should wait for the Backup and Rollback scripts to complete. If these scripts do not complete during this time, they will be aborted. The default is 3600 seconds (1 hour).
Undeploy	No	Enable or disable.

> **Note:** When sandboxing is enabled, you must specify a Base Path when you create a Code component. You cannot create a version of a release that contains a Code component whose Base Path is not specified (when sandboxing is enabled).

If the Application Deployment administrator removes a Base Directory that is used by a Code component in an existing release, you will not be able to create a new version of that release. You will, however, still be able to deploy existing versions of the release.

Using symbolic links

Application Deployment supports symbolic links in Code components. If you want to use symbolic links, note the following:

- Symbolic links are only for UNIX targets; they do not work on Windows operating systems.
- If the Source Directory is a symbolic link, SA will follow that link to find the files for the Code component.
- If a symbolic link is found *within* the directory structure to which the Source Directory points, that symbolic link is archived—it is not followed. When the files are installed on the target server, the symbolic link is created with its original value.
- Symbolic links are treated the same as files in terms of inclusions and exclusions.

Using filters

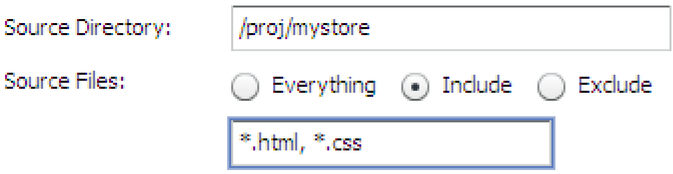
You can use a filter with the Include/Exclude options to specify which files are included in the Code component.

- For a full release, all files and directories that meet the filter criteria are included.
- For a delta release, only those files that meet the filter criteria and have a time-stamp later than the pertinent version are included. If a file has been deleted from the filesystem, it will not be included in the component package.

The filters recognize the `*` wildcard character. Application Deployment recursively searches the subdirectories of the Source Directory for files that match the patterns that you specify. For example:

- `*/src/*` matches all files with a directory named `src` in the path
- `Fred*` matches all files whose names begin with `Fred` and directories named `Fred` in the path

Be sure to separate your filter patterns with commas, as shown here:



Source Directory: /proj/mystore

Source Files: ○ Everything ● Include ○ Exclude

*.html, *.css

This pattern, for example, would match all files with either the `*.html` or `*.css` file name extensions, including the following:

```
/proj/mystore/index.html

/proj/mystore/products/dogs/dachshunds.html

/proj/mystore/main.css
```

Using a source code control System

CVS and Subversion are examples of source code control systems. Application Deployment integrates with most popular source code control systems.

In this case, the Source Directory refers to the location of the client copy of the source repository that exists on the SA Core server. You can use the Include/Exclude filters to specify which files to include in your component.

Using a source code control system source type provides the following benefits:

- Every time that you create a version of a release, the source code control system is used to refresh the files in the client copy. This way, you can ensure that the latest updates are included in the Code component.
- For delta releases, if a file is deleted from the source code control system, it will also be deleted from the target servers when the new version is deployed.

If you have multiple Code components that access the same source code control repository, be sure to specify a different Source Directory for each component. This prevents concurrency errors from occurring when the application is deployed.

Your Application Deployment administrator must integrate your source code control system before you can use this Source Type in your applications. See "Administering code component source types" on page 603 for more information.

Using a URL

When you create a Code component with the URL source type, you specify a URL that points to a single file that can be accessed by whatever protocol is specified in that URL (usually HTTP). For example, you can retrieve the latest nightly build of an application from the location where your automated build system places it.

If you use HTTPS protocol, you must ensure that the required certificates are installed in the correct location.

Script components

Script components are run on the target **during the deployment** of an application. For example, you might have two scripts: one that shuts down all applications before the deployment, and another that starts the applications up again after deployment.

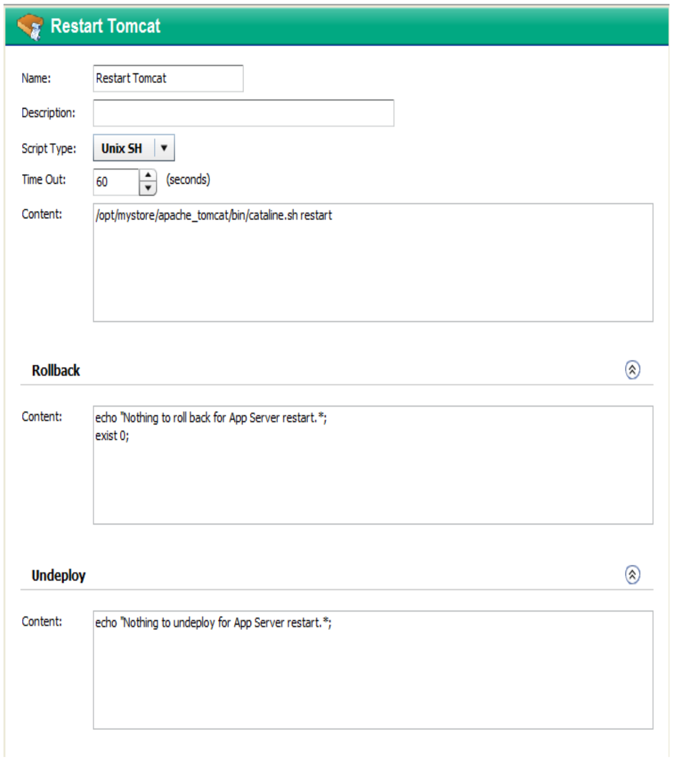
When you create a Script component, specify the following information:

Script Component Properties

Property	Required?	Purpose
Name	Yes	Name of the component. This name appears in the tier. It must be unique within the application.
Description	No	Field that you can use to document the component for other Application Deployment users.
Script Type	Yes	Type of script. The choices available depend on the platform required by the tier.
Time Out	Yes	Time (in seconds) that SA should wait for the script to finish running.
Content	Yes	Actual instructions that will run on the target during deployment.
Rollback Content	No	If you want to enable rollback or undeploy, you must provide explicit instructions for each scenario. Use the same scripting language that you use for the Content of the script itself.
Undeploy Content	No	

The following simple example restarts the Tomcat application server on the target.

Script component example for a UNIX target



Restart Tomcat

Name: Restart Tomcat

Description:

Script Type: Unix SH

Time Out: 60 (seconds)

Content:

```
/opt/mystore/apache_tomcat/bin/cataline.sh restart
```

Rollback

Content:

```
echo "Nothing to roll back for App Server restart.*;
exist 0;
```

Undeploy

Content:

```
echo "Nothing to undeploy for App Server restart.*;
```

You can use parameters in Script components. See "Parameters and special variables" on page 502 for more information.

Configuration File components

A Configuration File component creates a new configuration file on the target. This file contains **specific information** that the application requires. For example, you might need to create a configuration file that specifies the database to use for a particular application.

Configuration File components are most suitable for simple key-value types of files.

For complex parameterization or application-specific configuration, use an Application Configuration component instead (see "Application Configuration components" on page 479).

When you create a Configuration File component, you must specify the following information:

Configuration File component properties

Property	Required?	Purpose
Name	Yes	Name of the component. This name appears in the tier. It must be unique within the application.
Description	No	Description that appears in the reports.
Base Path	No	The Base Path is combined with the Destination File to form the Full Install Path—where the configuration file will be placed on the managed server. The Base Path and Full Install Path are only visible when Sandboxing is enabled (see Administering Application Settings). See the note immediately following this table.
Destination File	Yes	Name of the configuration file that will be created on the target server. If Sandboxing is enabled, the Base Path is combined with the Destination File to form the Full Install Path, where the configuration file will be created. If Sandboxing is not enabled, you must specify the full absolute path where the configuration file will be created on the target server. Do not specify a relative path. For example: `/tmp/webstore/config/db.cfg`
Full Install Path	Read-only	Base Path + Destination File (only visible when Sandboxing is enabled).

Configuration File component properties, continued

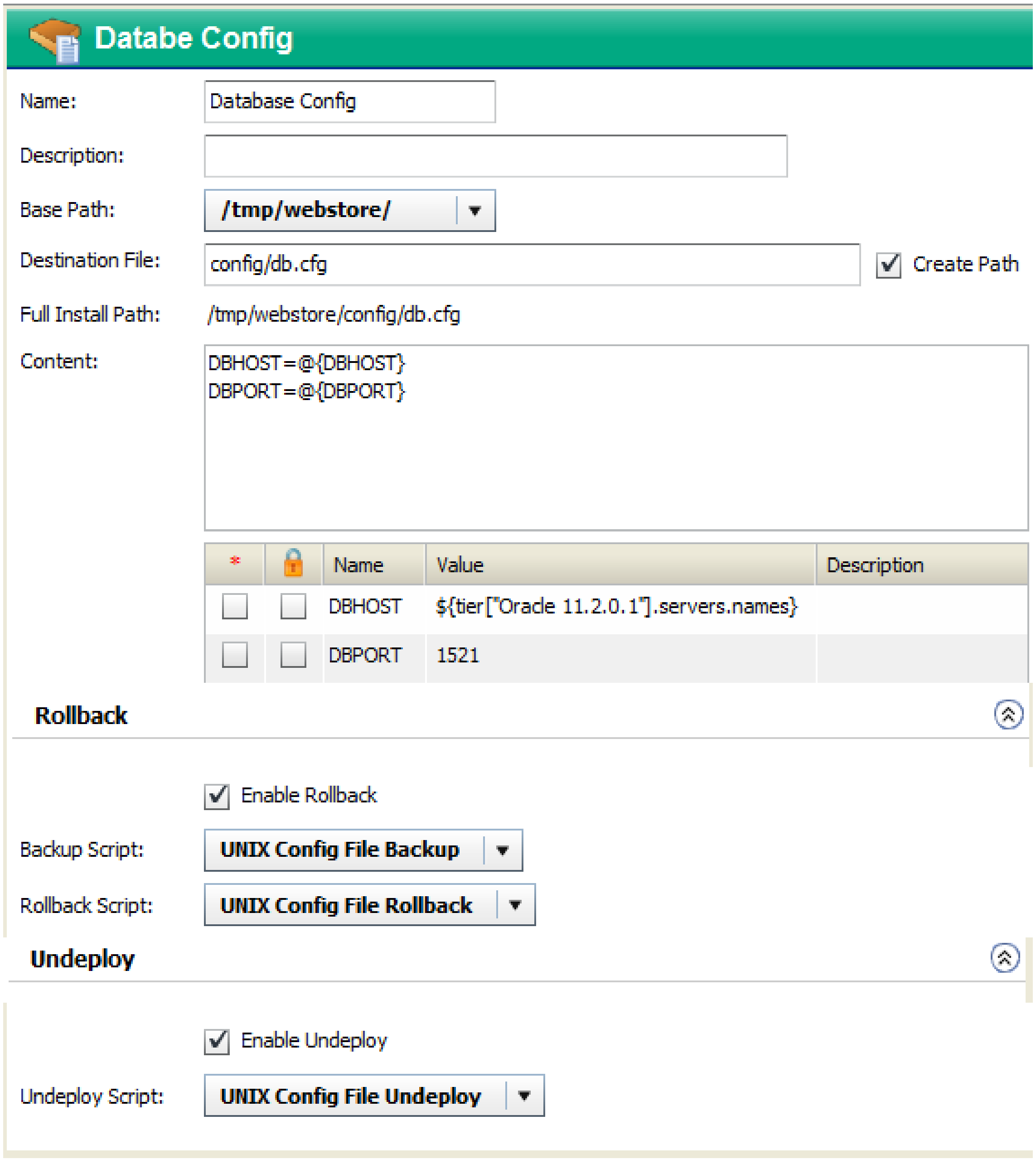
Property	Required?	Purpose
Create Path	No	If Create Path is selected, the directory hierarchy required to place the Destination File in the correct location will be created on the target server if that heirarchy does not already exist.
Content	Yes	Contents of the configuration file.
Rollback	n/a	The Backup, Rollback, and Undeploy scripts available depend on the platform of the tier. These scripts are maintained by your Application Deployment administrator. See "Administering scripts" on page 601 for more information.
Undeploy	n/a	

> **Note:** When sandboxing is enabled, you must specify a Base Path when you create a Configuration File component. You cannot create a version of a release that contains a Configuration File component whose Base Path is not specified (when sandboxing is enabled).

If the Application Deployment administrator removes a Base Directory that is used by a Configuration File component in an existing release, you will not be able to create a new version of that release. You will, however, still be able to deploy existing versions of the release.

Configuration file components can use parameters whose values are assigned at deployment time. See "Types of components" on page 470 for more information.

Configuration file component example for a UNIX target



Databe Config

Name: Database Config

Description:

Base Path: /tmp/webstore/

Destination File: config/db.cfg ☑ Create Path

Full Install Path: /tmp/webstore/config/db.cfg

Content:

```
DBHOST=@{DBHOST}
DBPORT=@{DBPORT}
```

*	🔒	Name	Value	Description
□	□	DBHOST	${tier["Oracle 11.2.0.1"].servers.names}	
□	□	DBPORT	1521	

Rollback

☑ Enable Rollback

Backup Script: UNIX Config File Backup

Rollback Script: UNIX Config File Rollback

Undeploy

☑ Enable Undeploy

Undeploy Script: UNIX Config File Undeploy

This simple example creates a file called `db.cfg`, which contains the database server host name and port. The values of the `DBHOST` and `DBPORT` parameters are assigned when the component is deployed to the target. The value of `DBPORT` is 1521. The value of DBHOST is a comma-separated list of the names of servers that are part of the Oracle 11.2.0.1 tier in the target.

Application Configuration components

SA enables you to manage configuration files from a central location and easily propagate changes and updates across multiple servers in your data center. You can manage a single configuration file (for example, the `/etc/hosts` file on UNIX systems) or multiple complex configuration files associated with an application (for example, the configuration files associated with a large business application, such as Web Logic or WebSphere).

The mechanism that SA uses to manage configuration files is called an **Application Configuration**. It consists of two types of elements: one or more templates, and a set of values.

Application Deployment enables you to create components that use existing Application Configurations from the SA Library. You can modify items in the value set at deployment time. For example, there might be an XML file that must contain information specific to the targets for the application deployment—for example, which server hosts the database.

When you create an Application Configuration component, specify the following information:

Application Configuration component properties

Property	Required?	Purpose
Name	Yes	Name of the component. This name appears in the tier. It must be unique within the application.
Description	No	Description that appears in the reports.
Application Configuration Name	Yes	These three properties are established in the Application Configuration in the SA Library. You cannot modify these properties in the Application Deployment user interface. If you have the appropriate permissions on the folder containing the Application Configuration, however, you can open the SA Application Configuration window and modify these properties there. To open this window, click the Name link for this Application Configuration.
Application Configuration Description	No	
Instance Name	Yes	
Parameters	No	Any items in the value set of the Application Configuration that you are allowed to modify at deployment time are listed in the table.

Application Configuration component properties, continued

Property	Required?	Purpose
		If multiple templates exist for a particular Application Configuration, all parameters that are editable from all templates will be displayed in the Application Deployment user interface.
Enable Rollback	n/a	You can either enable or disable rollback and undeploy for this Application Configuration. The actual rollback and undeploy behaviors are dictated in the Application Configuration itself.
Enable Undeploy	n/a	

The following example shows an Application Configuration for an application that is designed to be deployed over many different networks, each with a different database server. That application is designed to retrieve connection information from an XML configuration file, `dbinfo.xml`. Four values in `dbinfo.xml` are required for the application to be able to connect to the database:

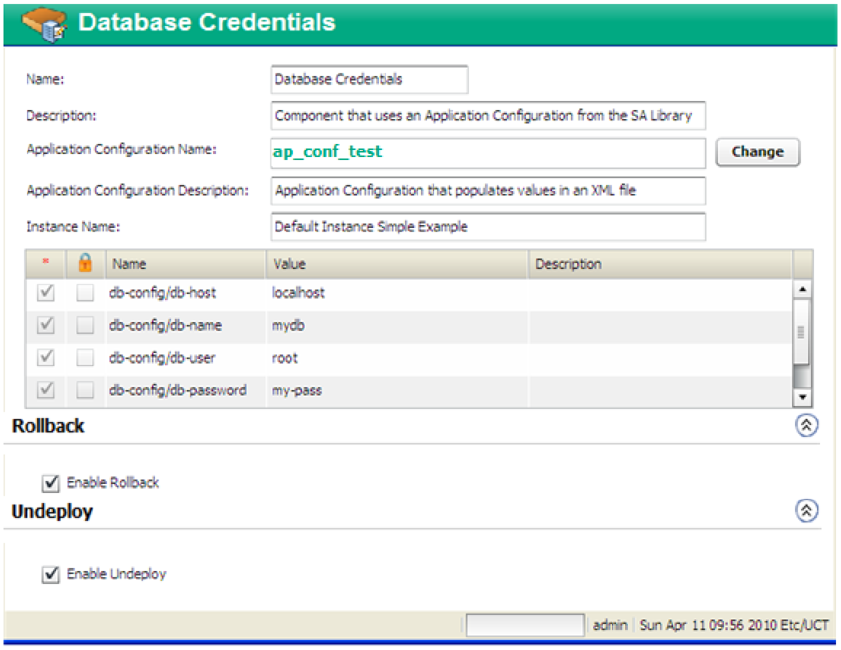
- **Host**: Host name of the server where the database is installed
- **Name**: Name of the database on the host server
- **User**: User name credentials used to open a connection to the database
- **Password**: Password necessary to open a connection to the database

Here is an example of the `dbinfo.xml` file:

```
<?xml version="1.0" ?>
<db-config>
<db-host>localhost</db-host>
<db-name>mydb</db-name>
<db-user>root</db-user>
<db-password>my-pass</db-password>
</db-config>
```

Here is a component based on this Application Configuration. Here, the `db-host` parameter will be modified at deployment time to use the name of the target.

Application Configuration component example



Database Credentials

Name:	Database Credentials
Description:	Component that uses an Application Configuration from the SA Library
Application Configuration Name:	ap_conf_test
Application Configuration Description:	Application Configuration that populates values in an XML file
Instance Name:	Default Instance Simple Example

*	🔒	Name	Value	Description
☑	□	db-config/db-host	localhost	
☑	□	db-config/db-name	mydb	
☑	□	db-config/db-user	root	
☑	□	db-config/db-password	my-pass	

Rollback

☑ Enable Rollback

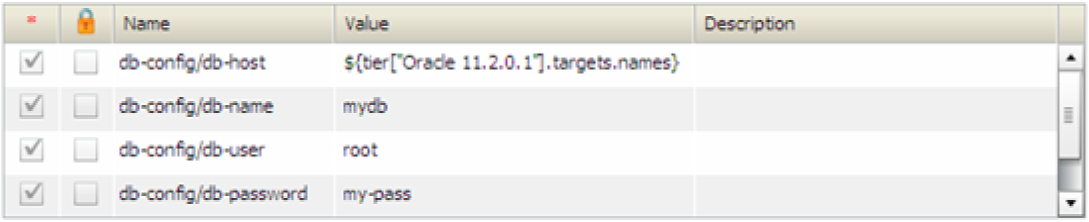
Undeploy

☑ Enable Undeploy

admin | Sun Apr 11 09:56 2010 Etc/UCT

To configure an Application Configuration component:

1. Create a new component of type Application Configuration.
2. Find and select the Application Configuration that you want to use from the SA Library.
3. To modify the value of a parameter, click the Value cell for that parameter in the table. You can specify a constant value or select a special variable to assign the value at deployment time, as shown here.

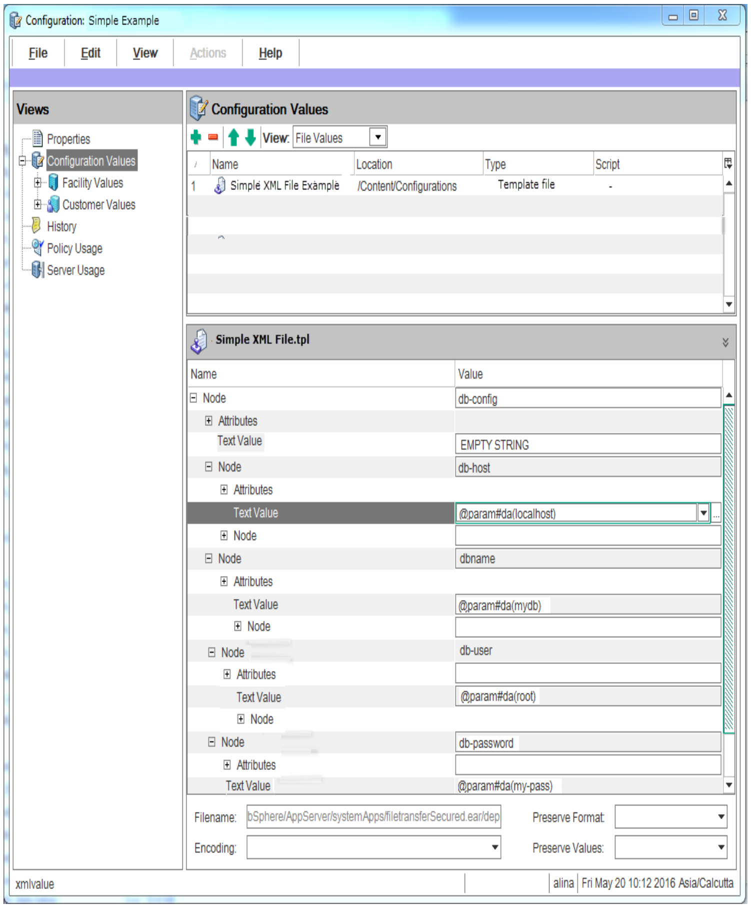
*	🔒	Name	Value	Description
☑	□	db-config/db-host	${tier["Oracle 11.2.0.1"].targets.names}	
☑	□	db-config/db-name	mydb	
☑	□	db-config/db-user	root	
☑	□	db-config/db-password	my-pass	

   For more information, see "Parameters and special variables" on page 502.

4. Enable or disable Rollback and Undeploy.

Be sure to save the component before you navigate to another screen.

To make an Application Configuration parameter available in the Application Deployment user interface, you must specify the Deployment Automation Value option for that parameter on the SA server, as shown here:



View: File Values

	Name	Location	Type	Script
1	Simple XML File Example	/Content/Configurations	Template file	-

Name	Value
⊟ Node	db-config
⊞ Attributes	
Text Value	EMPTY STRING
⊟ Node	db-host
⊞ Attributes	
Text Value	@param#da(localhost)
⊞ Node	
⊟ Node	dbname
⊞ Attributes	
Text Value	@param#da(mydb)
⊞ Node	
⊟ Node	db-user
⊞ Attributes	
Text Value	@param#da(root)
⊞ Node	
⊟ Node	db-password
⊞ Attributes	
Text Value	@param#da(my-pass)

Filename: bSphere/AppServer/systemApps/filetransferSecured.ear/dep

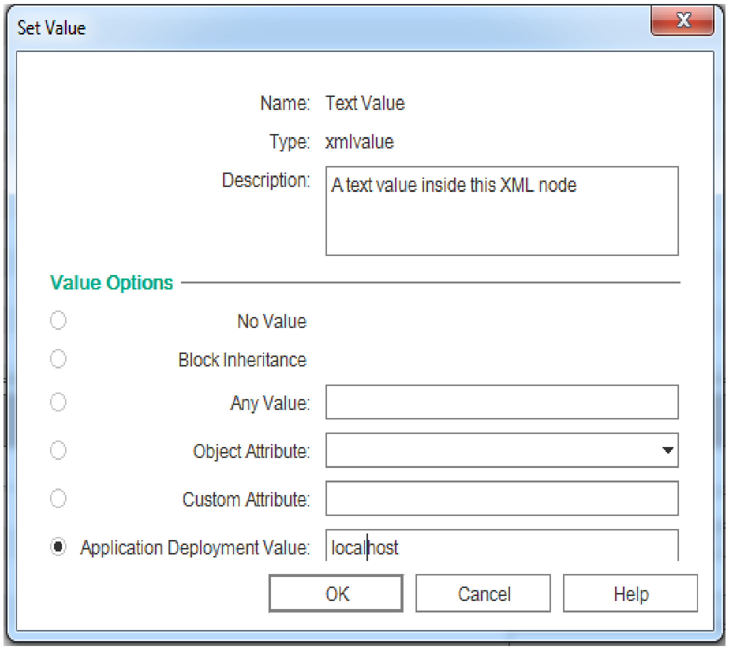
Encoding:

Preserve Format:

Preserve Values:

xmlvalue | alina | Fri May 20 10:12 2016 Asia/Calcutta

To do this, select the Value box for the parameter, and click the … button. The Set Value dialog opens:



Name: Text Value

Type: xmlvalue

Description: A text value inside this XML node

○ No Value

○ Block Inheritance

○ Any Value:

○ Object Attribute:

○ Custom Attribute:

● Application Deployment Value: localhost

Select the Deployment Automation Value option. If you want to provide a default value, type that value in the text box.

For more information about Application Configurations, see the SA 10.50 User Guide.

Software Policy components

Server Automation (SA) uses software policies to automate the process of installing software and configuring applications on managed servers. A Software Policy can contain packages, RPM packages, patches, Application Configurations, scripts, and server objects. It can also be associated with OS Sequences. After a Software Policy is created, it can be attached to servers or groups of servers. You can then assess which managed servers are in or out of compliance with a particular policy.

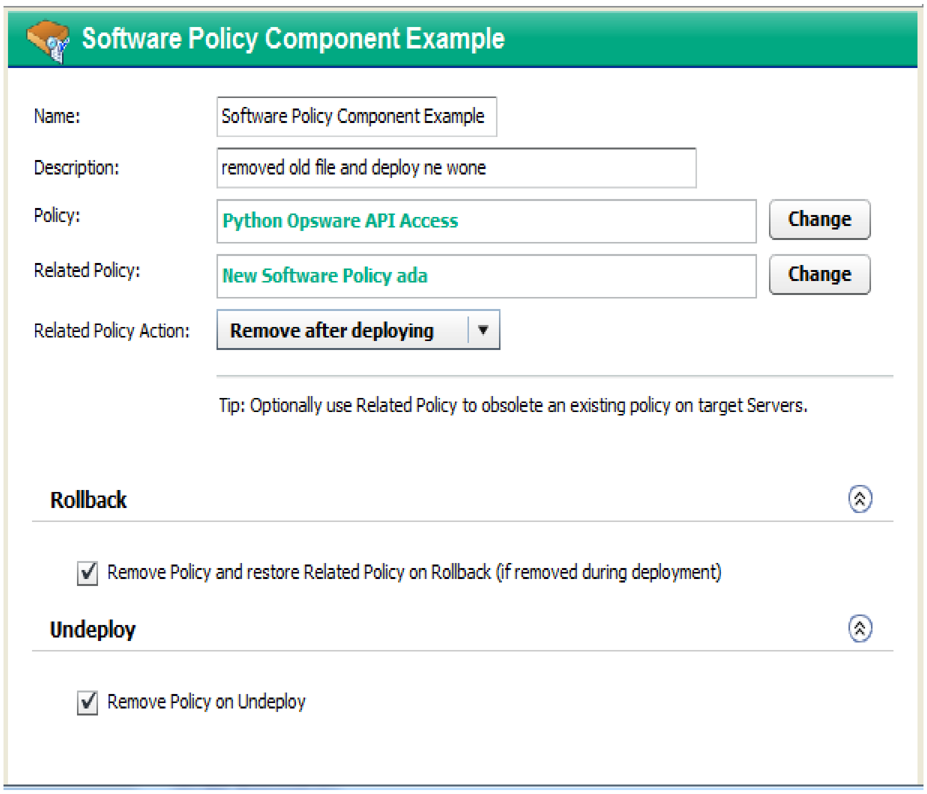
You can create application components that are based on software policies in the SA Library.

When you create a Software Policy component, specify the following information:

Software Policy component properties

Property	Required?	Purpose
Name	Yes	Name of the component. This name appears in the tier. It must be unique within the application.
Description	No	Description that appears in the reports.
Policy	Yes	Policy that will be attached and remediated on the target servers when the application is deployed.
Related Policy	No	Existing policy (already associated with a server) that is being replaced by the policy specified in the component.
Related Policy Action	If Related Policy is specified	Remove after deploying (default), remove before deploying, or do not remove. If you specify Do Not Remove, the Related Policy will only be used during rollback.
Rollback	n/a	If you select the Rollback box, the following things happen in the event that the application deployment is rolled back: The Software Policy specified in the Policy box is removed. If the policy specified in the Related Policy box was removed during deployment, this policy is restored.
Undeploy	n/a	If you select the Undeploy box, the Software Policy specified in the Policy box is removed when this application is undeployed.

Software Policy component example



Software Policy Component Example

Name: Software Policy Component Example

Description: removed old file and deploy ne wone

Policy: Python Opsware API Access

Related Policy: New Software Policy ada

Related Policy Action: Remove after deploying

Tip: Optionally use Related Policy to obsolete an existing policy on target Servers.

Rollback

☑ Remove Policy and restore Related Policy on Rollback (if removed during deployment)

Undeploy

☑ Remove Policy on Undeploy

To configure a Software Policy component:

1. Create a new component of type Software Policy.
2. Find and select the Policy and Related Policy that you want to use from the SA Library.
3. Select the Related Policy Action that you want.
4. Specify the Rollback and Undeploy behaviors.

Be sure to save the component before you navigate to another screen.

For more information about software policies, see the “Creating and Managing Software Policies” section in the SAonline help.

Package components

Applications in the SA Software Repository are organized in packages. To install a package in SA, you would add that package to a Software Policy, attach that policy to one or more managed servers, and then perform a server remediation.

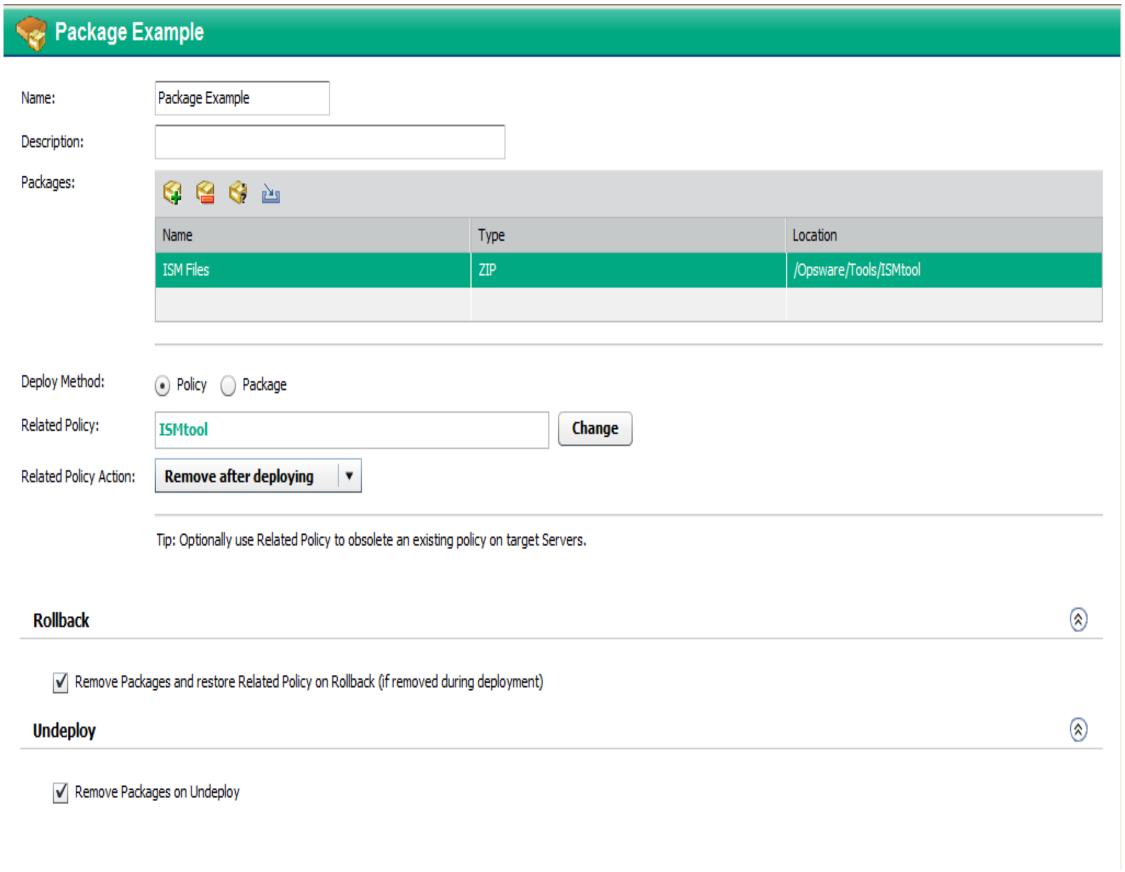
You can create Application Deployment components that include one or more packages from the SA Library and an optional related policy. You can also create a new package “on the fly” and import that package into the SA Library. You can choose between attaching a policy—and using the normal SA server remediation workflow—or simply installing the package without attaching a policy.

When you create a Package component, specify the following information:

Package component properties

Property	Required?	Purpose
Name	Yes	Name of the component. This name appears in the tier. It must be unique within the application.
Description	No	Description that appears in the reports.
Packages	Yes	List of packages from the SA Library.
Deploy Method	Yes	When you create a new version, Application Deployment creates a package containing the packages specified and any scripts required. Policy – Application Deployment also creates a policy when it creates a new version. When the version is deployed, it will attach and remediate this policy on each target server. Package – When the version is deployed, Application Deployment will perform an “ad hoc” install of the packages on each target server.
Related Policy Related Packages	No	Existing policies or packages that are being replaced by the packages specified in the component.
Related Packages Action	If Related Packages are specified	Remove after deploying (default), remove before deploying, or do not remove. If you specify Do Not Remove, the Related Packages will only be used during rollback.
Rollback	n/a	If you select the Rollback box, the following things happen in the event that the application deployment is rolled back: The packages are removed. If the specified Related Policy was removed, it is restored. The specified Related Packages are always restored.
Undeploy	n/a	If you select the Undeploy box, the packages listed in the Packages table are removed when this application is undeployed.

Package component example



Package Example

Name: Package Example

Description:

Packages:

Name	Type	Location
ISM Files	ZIP	/Opsware/Tools/ISMtool

Deploy Method: ● Policy ○ Package

Related Policy: ISMtool

Related Policy Action: Remove after deploying ▼

Tip: Optionally use Related Policy to obsolete an existing policy on target Servers.

Rollback

☑ Remove Packages and restore Related Policy on Rollback (if removed during deployment)

Undeploy

☑ Remove Packages on Undeploy

You can use the following tools to specify the packages in a component:

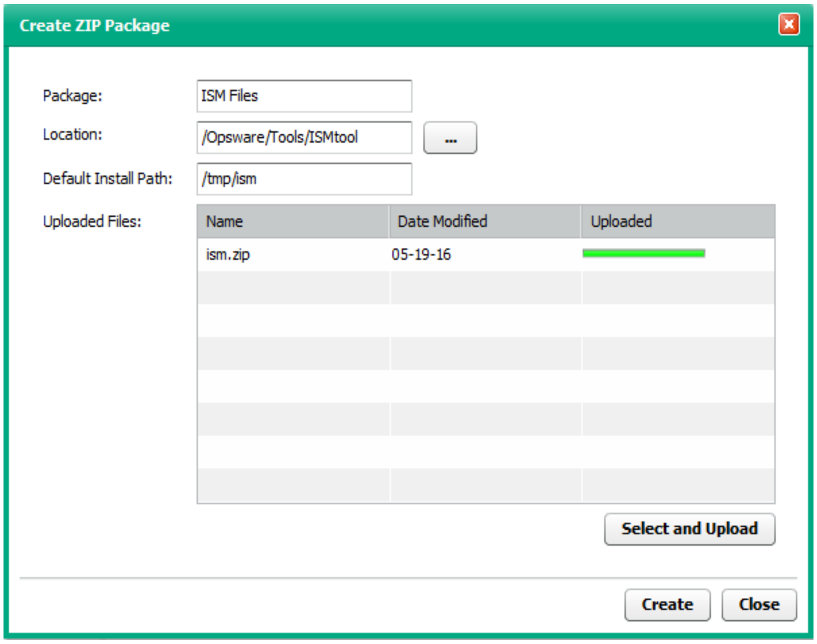
- Click the “Add Package” button to select a package from the SA Library. This opens the Select Package dialog. See "Working with components" on page 493.
- Use wildcard characters to select all packages that match a search string. To do this, follow these steps:
  a. Select a package in your Packages table.

  b. Click the “Wildcard Package Name” button. The Set Wildcard dialog opens.

  c. Modify the text in the Name box to create your search string using * and ? wildcard characters.

  d. Click **OK**.

  When you create a new version, Application Deployment will include the newest package in the same location (folder) in the SA Library that matches your search string and package type.
- Click **Create ZIP Package from Files** create a new ZIP package. This opens the Create ZIP Package dialog:



Package: ISM Files

Location: /Opsware/Tools/ISMtool

Default Install Path: /tmp/ism

Uploaded Files:

Name	Date Modified	Uploaded
ism.zip	05-19-16	🟩

The Location is the folder in the SA Library where the new package will reside.

The Default Install Path is the location on the target where it will be installed when the application is deployed.

To create a new package, follow these steps:

1. Click **Select and Upload**.
2. Browse to the file (or files) on your local system that you want to include in the package.
3. Click **Create** to create the ZIP package and import it into the SA Library.

You can now deploy your package as you would any other package in the SA Library.

To configure a Package component:

1. Create a new component of type Package.
2. Use the **Add Package** tool or the **Import Package** tool to specify the Packages that you want to include.
3. Specify the Deploy Method (see "Software Policy component properties" on page 483).
4. Use the **Add Package** tool or the **Import Package** tool to specify the Related Packages that you want to use, and select the **Related Policy Action**.
5. Specify the Rollback and Undeploy behaviors.

Be sure to save the component before you navigate to another screen.

For more information about software policies, see “Managing Software Packages” in the SA online help.

OO flow components

This type of component initiates an **HPE Operations Orchestration (OO)** workflow. This is useful if you want to integrate with other applications. For example:

- Issuing an HPE Service Manager trouble ticket indicating that a certain group of servers was updated
- Configuring a load balancer to know about the deployment
- Interacting with an application monitoring system

If your SA Core server is configured to integrate with an HPE OO server, you can create Application Deployment components that include an HPE OO workflow.

You can also associate HPE OO workflows with specific environments. Using this strategy, you can create the following type of scenario:

1. At the environment level, initiate a “pre-deployment” OO workflow.
2. Deploy the application.
3. At the environment level, initiate a “post-deployment” OO workflow.

For example: disable monitoring for 10 minutes®deploy the application®re-enable monitoring.

In a QA environment, you could run a smoke test (i.e., validation) after each nightly build. Then, you might want to email the product owner to see if the test result is acceptable. If an email response is received, you can deploy the new build.

See "Administering environments" on page 596 for more information.

When you create an OO Flow component, specify the following information:

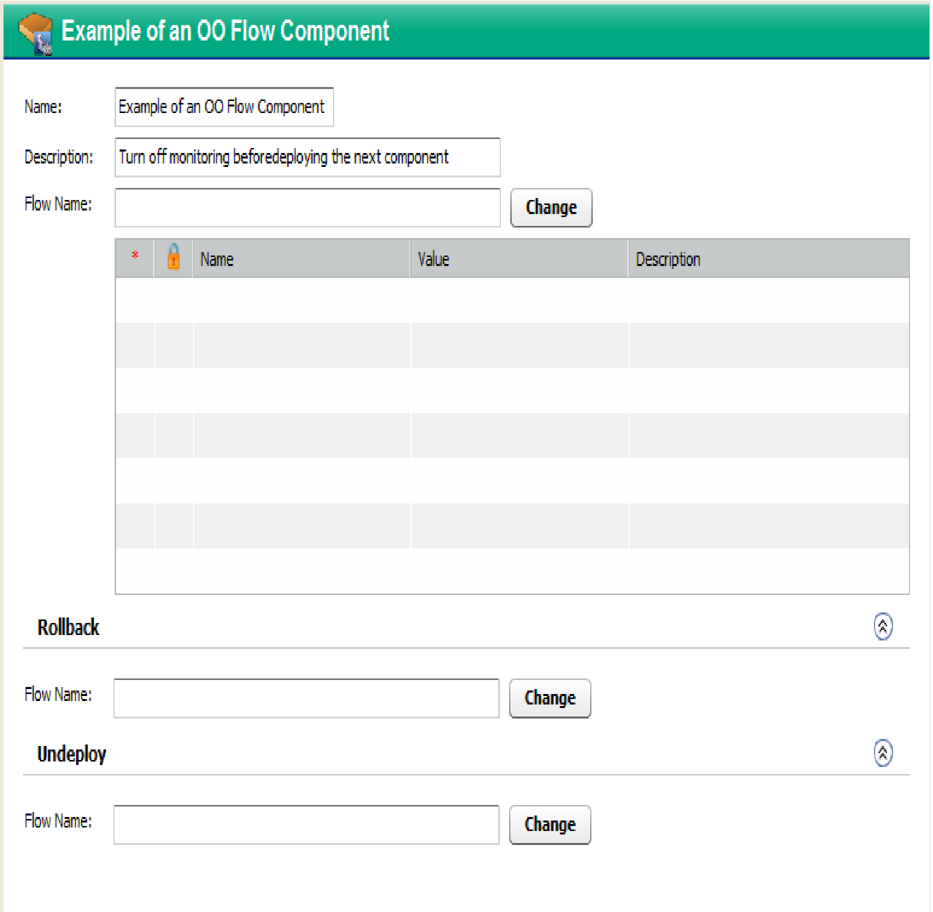
OO Flow component properties

Property	Required?	Purpose
Name	Yes	Name of the component. This name appears in the tier. It must be unique within the application.
Description	No	Description that appears in the reports.
Flow Name	Yes	Reference to an HPE OO workflow.

OO Flow component properties, continued

Property	Required?	Purpose
Flow Parameters	Depends on flow selected	Parameters required by the selected HPE OO workflow. Note that parameters with identical names carry across the functional, rollback, and undeploy sections.
Rollback Flow	No	HPE OO workflow to use for rollback in the event of a deployment failure.
Undeploy Flow	No	HPE OO workflow to use in the event that an application is undeployed.

OO flow component example



Example of an OO Flow Component

Name: Example of an OO Flow Component

Description: Turn off monitoring beforedeploying the next component

Flow Name:

To configure an OO flow component:

1. Create a new component of type OO Flow.
2. Select an HPE OO workflow from the HPE OO Library.

If you are searching for an OO Flow, you can dramatically speed up your search by specifying a full word in the text box. The OO Flow search only matches full words. For example, “start” does not match “restart” in this context.

3. To modify the value of a parameter, click the Value cell for that parameter in the table. You can specify a constant value or select a special variable to assign the value at deployment time (see "Parameters and special variables" on page 502).
4. Specify the Rollback and Undeploy behaviors.

Be sure to save the component before you navigate to another screen.

Windows Registry components

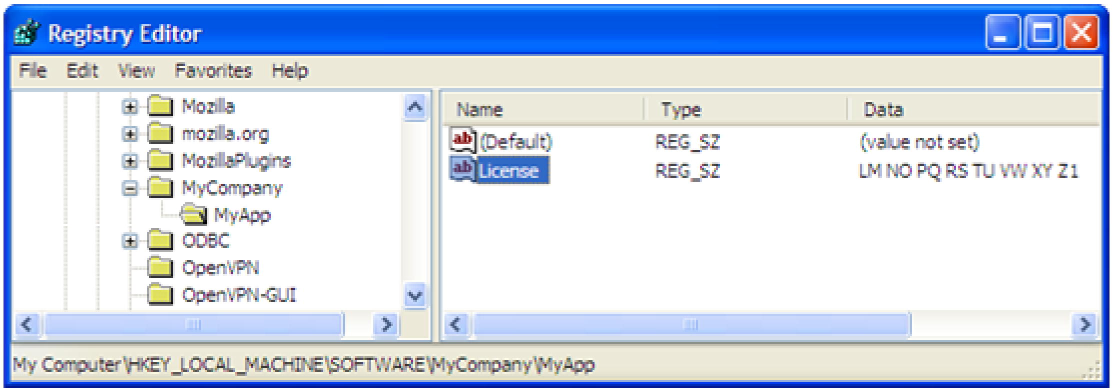
Use this type of component to add or delete **Windows Registry** keys or values. This type of component is only available for Windows applications.

When you create a Windows Registry component, specify the following information:

Windows Registry component properties

Property	Required?	Purpose
Name	Yes	Name of the component. This name appears in the tier. It must be unique within the application.
Description	No	Description that appears in the reports.
Key	Yes	Registry key that you want to modify.
Values	Yes	Values to add or delete from the specified key. Values are not required if you are deleting the key.
Backup Script	If rollback is enabled	The Backup, Rollback, and Undeploy scripts are maintained by your Application Deployment administrator. See "Administering scripts" on page 601 for more information.
Rollback Script		
Undeploy Script	If undeploy is enabled	
Undeploy Action	If undeploy is enabled	Delete either the entire key and any values or just the values that were added.

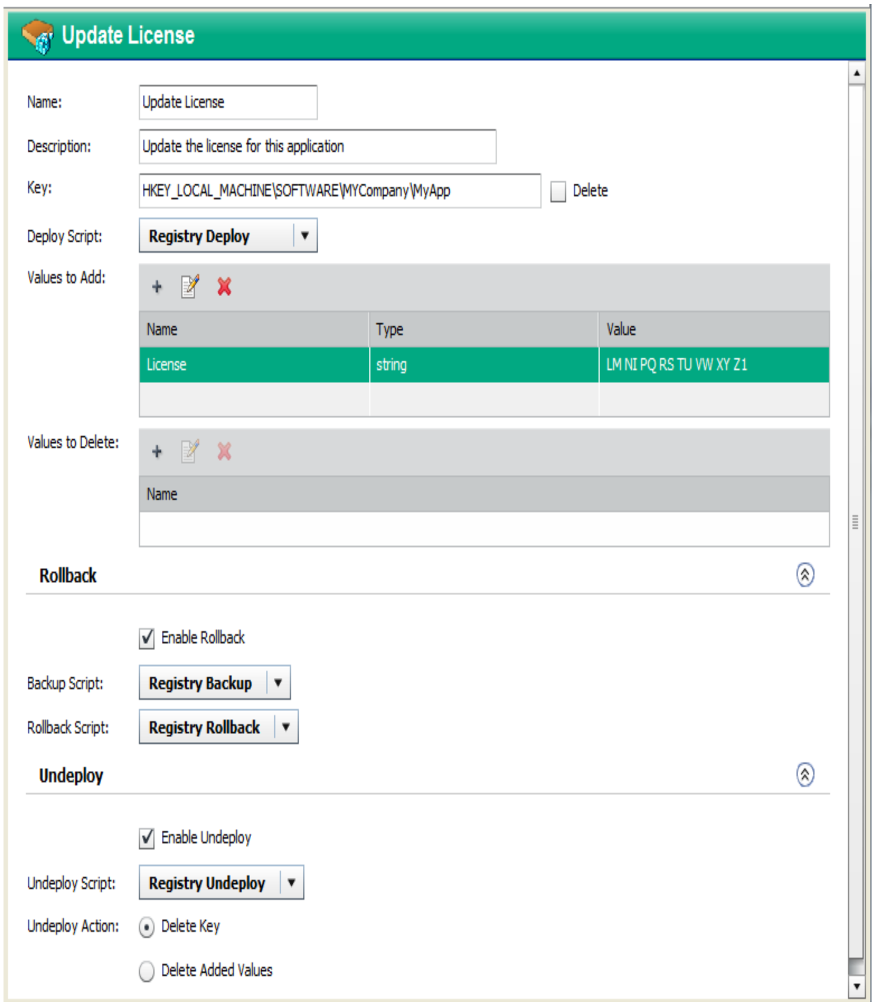
The following Registry component adds this License key value to the registry:



Name	Type	Data
(Default)	REG_SZ	(value not set)
License	REG_SZ	LM NO PQ RS TU VW XY Z1

My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\MyCompany\MyApp

Example of a Windows Registry component



Name:	Update License
Description:	Update the license for this application
Key:	HKEY_LOCAL_MACHINE\SOFTWARE\MYCompany\MyApp □ Delete
Deploy Script:	Registry Deploy

Values to Add:

Name	Type	Value
License	string	LM NI PQ RS TU VW XY Z1

Values to Delete:

Name

Rollback

☑ Enable Rollback

Backup Script:	Registry Backup
Rollback Script:	Registry Rollback

Undeploy

☑ Enable Undeploy

Undeploy Script:	Registry Undeploy
Undeploy Action:	● Delete Key ○ Delete Added Values

It is important to specify the Key name and Values in the proper format. Note the following:

- The Key field must begin with a valid hive name. The following hive names are valid: HKEY_CLASSES_ROOT, HKCR, HKEY_CURRENT_USER, HKCU, HKEY_LOCAL_MACHINE, HKLM, HKEY_USERS, HKU, HKEY_CURRENT_CONFIG, HKEY_PERFORMANCE_DATA, and HKEY_DYN_DATA.
- Only back-slashes can be used as key component separators. For example:
- HKEY_LOCAL_MACHINE\SOFTWARE\Hewlett-Packard\HP Virtual Rooms 8.0
- The Value field for a Binary type key value must be specified as a series of space-separated two digit hexadecimal numbers. For example: 01 A2 B3 C4
- The Value field for a Dword type key value must be specified as either a hexadecimal number starting with 0x or a positive decimal number.

To configure a Windows Registry component:

1. Create a new component of type Windows Registry.
2. To add a new key value, follow these steps.
   a. Click **Add** in the Values to Add table.
   b. In the Add Value dialog, specify the **Name**, **Type**, and **Value**.

      You can type the **Value**, or you can select a special variable whose value is assigned at deployment time (see "Types of components" on page 470).
   c. Click **OK**.
3. To delete an existing key value, follow these steps.
   a. Click **Add** in the Values to Delete table.
   b. In the Add Value to Delete dialog, specify its **Name**.
   c. Click **OK**.
4. To modify a value in either table, select that value, and click **Edit Properties**.
5. To remove a value from either table, select that value, and click **Delete**.
6. Specify the Rollback and Undeploy behaviors.

Be sure to save the component before you navigate to another screen.

Working with components

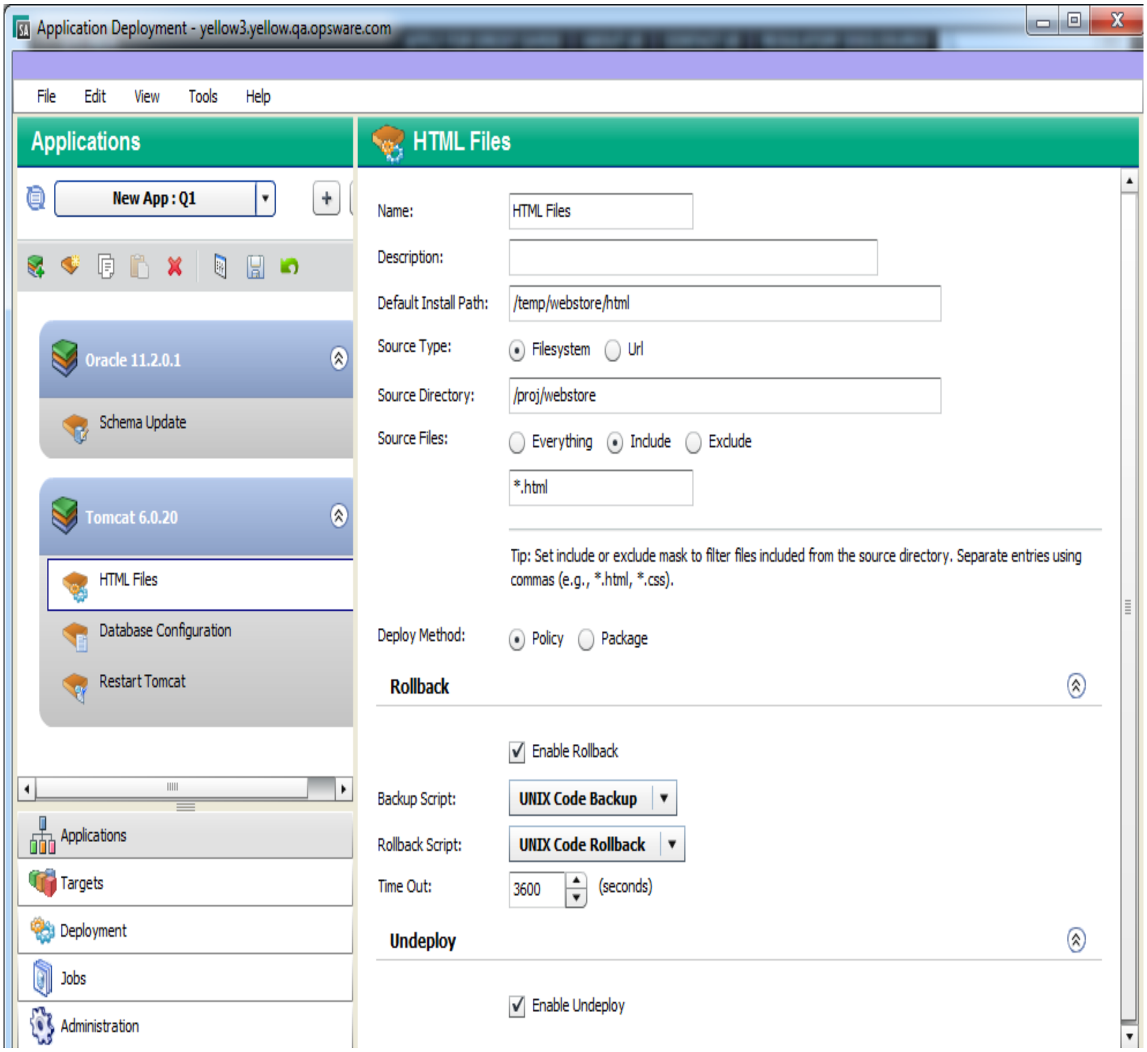
Components represent the **content** of an application. Each component deploys a part of the application.

Components are configured using the right pane of the Applications screen. Components belong to specific tiers—which represent the **structure** of an application.

Although the properties vary depending on the type of component that you are configuring, the process for creating and accessing components is similar.

You can add and modify components only if you have permission to edit the application.

Example of a code component



Name: HTML Files

Description:

Default Install Path: /temp/webstore/html

Source Type: ● Filesystem ○ Url

Source Directory: /proj/webstore

Source Files: ○ Everything ● Include ○ Exclude

*.html

Tip: Set include or exclude mask to filter files included from the source directory. Separate entries using commas (e.g., *.html, *.css).

Deploy Method: ● Policy ○ Package

Rollback

☑ Enable Rollback

Backup Script: UNIX Code Backup

Rollback Script: UNIX Code Rollback

Time Out: 3600 (seconds)

Undeploy

☑ Enable Undeploy

All components include three sets of properties:

The **functional** properties (located at the top of the pane) uniquely identify the component, specify its behavior, and tell Application Deployment where to find any items that this component includes.

The **rollback** properties tell Application Deployment what to do with this component in the event that a deployment fails and an orderly rollback is required.

For certain types of components, you must specify a rollback action if you enable rollback. This is either a script, a flow, or a set of explicit instructions, depending on the component type. You may also need to specify a backup action. In this case, the backup action is executed before the component is deployed; the rollback action restores the backed up items if the deployment fails.

The **undeploy** properties tell Application Deployment how to uninstall this component in the event that an instruction to undeploy the application is given.

Rollback and undeploy scripts are managed by the Application Deployment administrator (see "Administering scripts" on page 601).

Creating a new component

The New Component dialog is used to create a new application component. You can create new components for applications that you have permission to edit (see "Application permissions" on page 589).

A component is associated with a specific tier. You can only create a new component when a tier is selected. The platform to which a tier pertains determines the types of components that you can create.

To create a new component:

1. Select the tier that will contain the new component. There are two ways to select a tier:
   - Click the tier name.
   - Select any component within that tier.
2. Click **New Component tool bar**.
3. In the New Component dialog, specify the **Component Name** and **Component Type**. Component names must be unique within an application.
4. Click **OK**.
5. Fill in the appropriate values for the properties of your new component. See "Types of components" on page 470 for details.
6. Click **Save** to save your changes.

If you attempt to leave the Applications screen without saving your changes, you will be reminded to save or discard them.

Modifying an existing component

If you have permission to edit an application, you can modify its components (see "Application Deployment permissions" on page 589). When you select a component in the left pane, you can modify its properties in the right pane. The types of modifications that you can make depend on the type of component.

To modify an existing component:

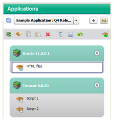
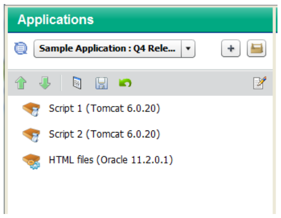
1. Select that component in the tier.
2. Modify its properties using the tools in the right pane.
3. Click **Save** to save your changes.

If you attempt to leave the Applications screen without saving your changes, you will be reminded to save or discard them.

Changing the deployment order of components

If you have permission to edit an application, you can change the order in which its components are deployed to targets (see "Application permissions" on page 589). To do this, you must change the left pane from the tier view (the default) to the deployment order view, as shown here.

Component views

Tier View	Deployment Order View
	 Applications Sample Application : Q4 Rele... Script 1 (Tomcat 6.0.20) Script 2 (Tomcat 6.0.20) HTML files (Oracle 11.2.0.1)

For example, you might need to start an application after its files have been deployed. In this case, you would make sure the Script Component that starts the application comes later in the deployment order than the Code Component that deploys the files.

Unless you change their order, components are deployed in the order in which they were created.

To modify the component deployment order:

1. Click **Edit Component Deployment Order**.

   The left pane changes to show all the components associated with the current application. They are listed in the order in which they will be deployed:

2. Select a component, and use the green arrows (⬆ and ⬇) to move the component up or down in the list. In this view, the tier is shown in parentheses after the component name. When you move components using the arrow buttons, they remain in their original tiers.

3. To return to the tier view, click the “Edit Properties” button: 📝

In the tier view, you can drag and drop a component from one tier to another. Note that certain types of components can only be used in tiers designed for a particular operating system. For example, a Registry component can only be used in a Windows tier.

If you drag and drop a component into a different tier, be sure to check the deployment order. By default, components are placed at the end of the tier.

Copying and pasting a component

You can copy an existing component and then paste that component into a compatible tier in the same application or another application. Compatible tiers run on the same OS platform.

When you copy and paste a component, all the properties of the existing component are copied into the new component.

You can only copy and paste one component at a time, and you can only copy and paste components in applications that you have permission to edit (see "Application permissions" on page 589).

To copy and paste a component:

1. Select the component that you want to copy.

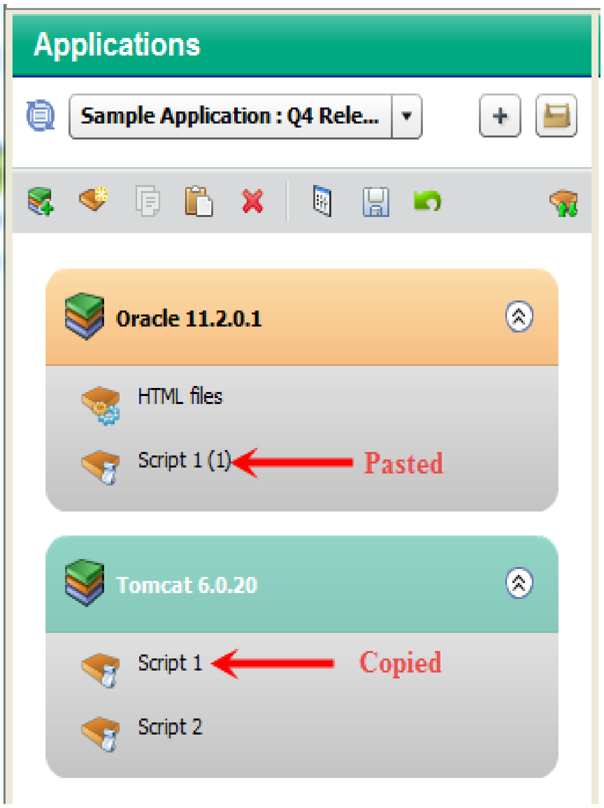
   This activates the tier that contains this component and makes the **Copy** button 📋 available on the tool bar.

   If the **Copy** button is not available, you do not have permission to edit this application.

2. Click **Copy** on the tool bar.

3. *Optional*: If you want to paste the component into a different application, open that application for editing.
4. Select the tier where you want to paste the component.

   If you are pasting into the same tier where you performed the copy, you do not need to select it.
5. Click **Paste** on the tool bar.The name of the pasted component will have a number appended to it as shown in the following figure:

   If **Paste** is not available, you do not have permission to edit this application.
6. Click **Save** to save your changes.

Moving a component

You can use “drag and drop” to move a component from one tier to another compatible tier in the same application. Compatible tiers run on the same OS platform

You can only move one component at a time, and you can only move components in applications that you have permission to edit (see "Application permissions" on page 589).

To move a component using drag and drop:

1. Drag the component that you want to move into the component area in a compatible tier.

   A green plus sign ⊕ appears if the new tier is compatible with the original tier.
   A red X ⊗ appears if the new tier is incompatible—or if you attempt to drop the component into the same tier.

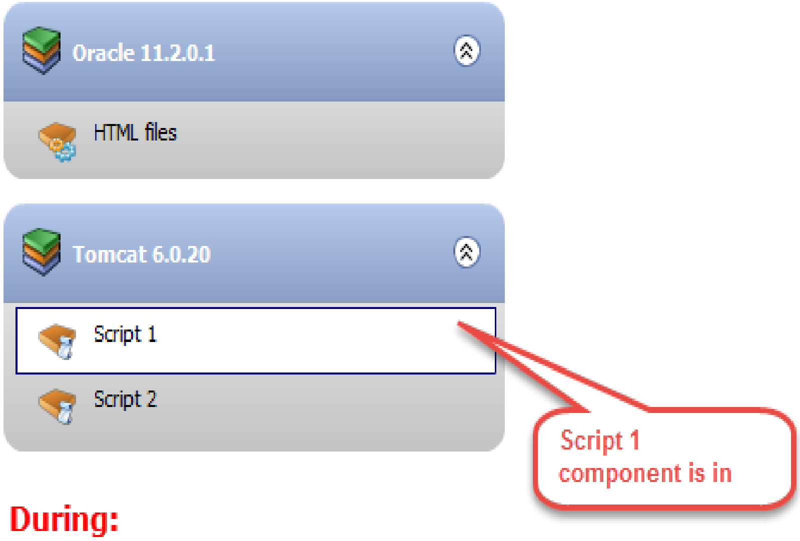
2. Drop the component into the new tier.

3. Click **Save** to save your changes.

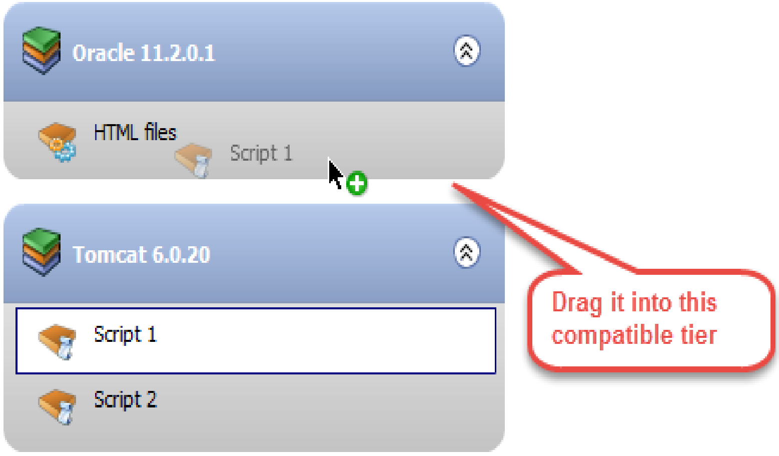
The following example shows the HTML Files component being moved from the Tomcat 6.0.20 tier into the Oracle 11.2.0.1 tier.

Example of moving a component to a different tier

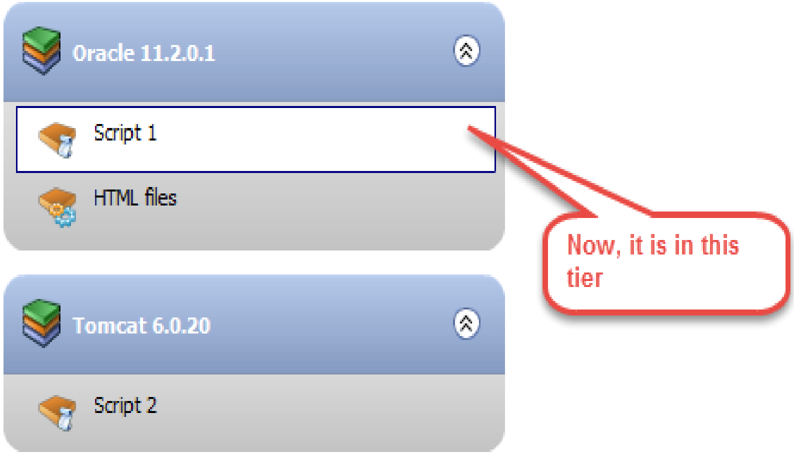
Before:



During:



After:



Deleting a component

You can delete components that belong to applications that you have permission to edit (see "Application permissions" on page 589).

To delete a component:

1. Select the component.
2. Click **Delete Selected Item** on the tool bar.
3. Click **Yes** to confirm.

Note that you can only delete one component at a time.

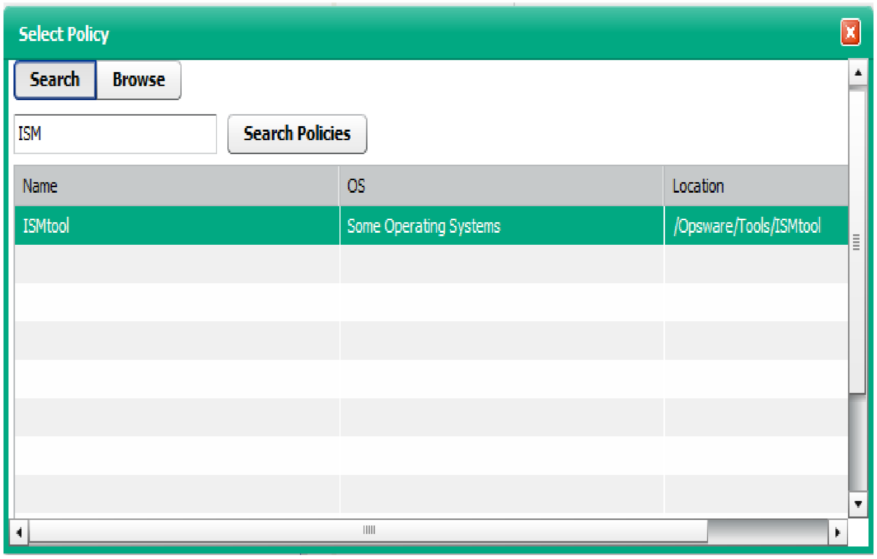
Finding and selecting an item in the library

The following information applies to the following types of components:

- "Application Configuration components" on page 479
- "Working with components" on page 493
- "Working with components" on page 493
- "Working with components" on page 493

In each case, you use a dialog like this one to search the pertinent library (SA OO) for a particular item.

Library Item Selection window – Search tab

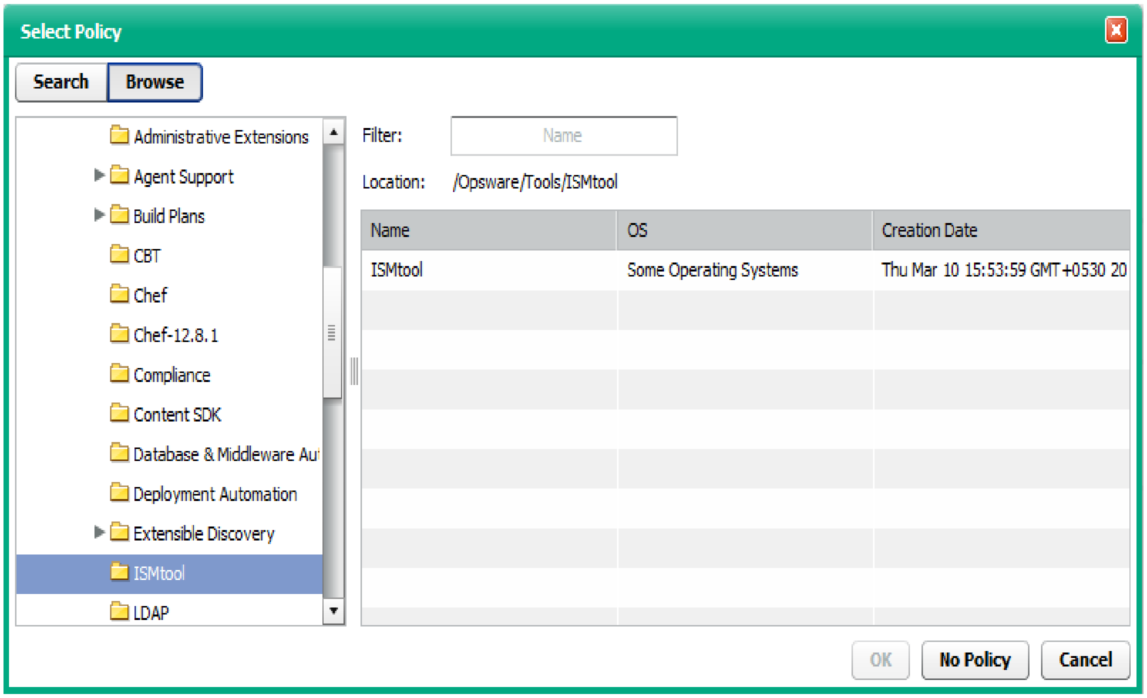


ISM

Name	OS	Location
ISMtool	Some Operating Systems	/Opsware/Tools/ISMtool

In this case, we are searching the SA Library for a policy whose name contains “ISM.” You can search the library by using the table view, as shown above, or you can browse through the folder hierarchy of the library.

Library Item Selection window – Browse tab



Filter:

Location: /Opsware/Tools/ISMtool

Name	OS	Creation Date
ISMtool	Some Operating Systems	Thu Mar 10 15:53:59 GMT+0530 20

In this case, the search has matched all policies in the `/Opsware/Tools/ISMtool` folder that have “Linux” in their names.

The following instructions assume that you have selected a Release on the Applications screen. The placeholder “Item” refers to one of the library items listed above.

To find and select an item in a library:

1. In the left pane, select one of the types of components for editing (or create a new component):
2. Click **Change** for the item (or the icon, if you are working with a Package component).

   The Select Item dialog opens.
3. *Optional*: If you are specifying a Package, select the type of package from the list: RPM, ZIP, MSI (available for Windows tiers only), or All Types.
4. T o search using the table view, follow these steps:
   a. In the text box, type the search string that you want to match. To list all the items in the library, leave the text box empty.

      If you are searching for an OO Flow, you can dramatically speed up your search by specifying a full word in the text box. The search only matches full words. For example, “start” does not match “restart” in this context.

   b. Click **Search Item**.

   To search using the folder hierarchy, follow these steps:

   a. Click the **Browse** button. You will see all the folders that you have permission to view.

   b. *Optional*: Specify a filter.
5. In the table, select the item that you want.
6. Click **OK**.

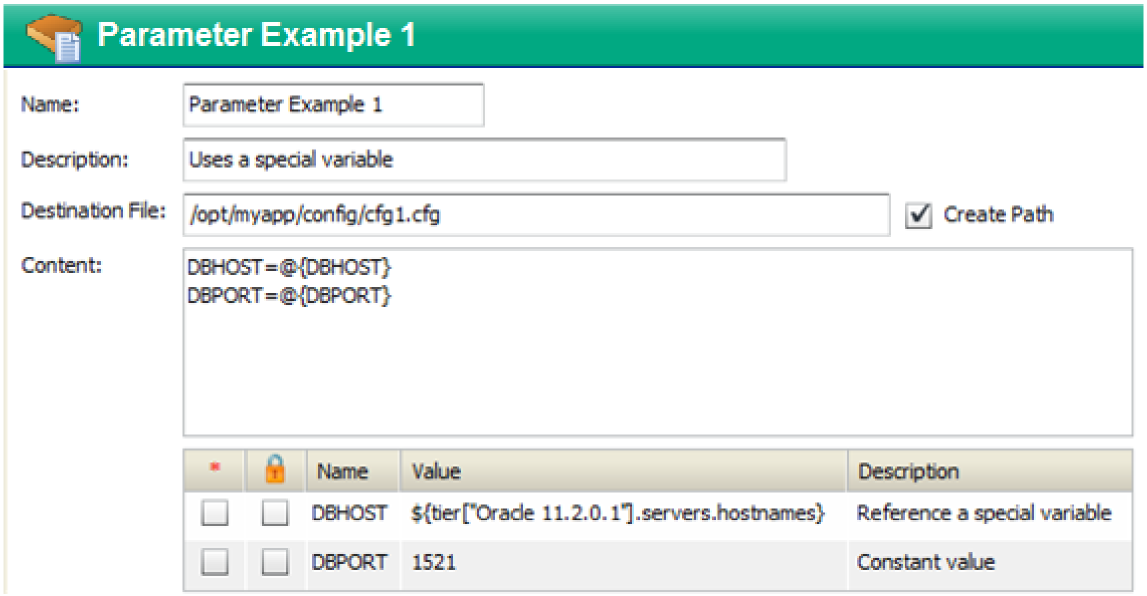
If you want to remove a library item from an existing component, click the **No Item** button. This button is available for Software Policyand OO Flow components. To remove a previously specified policy, for example, you would click **No Policy**.

Parameters and special variables

In certain types of components, you can use **component parameters** to represent information that may differ depending on the application, release, version, target, tier, or environment.

In the following Configuration File component, the parameters `DBHOST` and `DBPORT` are used to represent the database server host names and database port, respectively.

Example of parameters



Parameter Example 1

Name:	Parameter Example 1	
Description:	Uses a special variable	
Destination File:	/opt/myapp/config/cfg1.cfg	☑ Create Path

Content:

```
DBHOST=@{DBHOST}
DBPORT=@{DBPORT}
```

*	🔒	Name	Value	Description
□	□	DBHOST	${tier["Oracle 11.2.0.1"].servers.hostnames}	Reference a special variable
□	□	DBPORT	1521	Constant value

The notation for a parameter reference is as follows: `@{parameter_name}`

When you reference a parameter by using this notation in the Content box, the parameter definition appears in the table immediately below the Content box.

- The Name of the parameter is what appears between the curly braces in the Content box.
- A component parameter name may not contain the following characters: space, \, @, {, or }
- If the box in the * (required) column is selected, the parameter is required to have a value when the component is deployed.
- If the box in the 🔒 (encrypted) column is selected, the parameter is encrypted, and its value is not displayed anywhere in the Application Deployment user interface.

The value of a parameter can be a simple number or text—as in the `DBPORT` parameter in this example—or it can be a **variable** whose value is assigned at deployment time. The `DBHOST` parameter in the example references a variable that represents a comma-separated list of the host names of the servers that are part of the Oracle 11.2.0.1 tier in the specified target.

There are three types of variables that you can use to specify parameter values: special variables, release parameters, and global parameters. Each has a specific purpose and function (see "Types of variables" on the next page).

When you specify the value of a parameter in the component definition, you are specifying the default value for that parameter. You can modify that value at deployment time. This is useful, for example, if you want to override the default value for a specific target or environment (see "Customizing deployment parameters" on page 544).

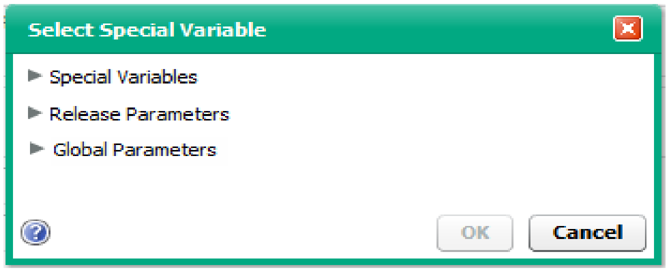
For information about creating, managing, and referencing parameters, see "Working with parameters and variables" on page 509. For information about rules and constraints related to parameters, see "Special considerations" on page 514.

Types of variables

There are three types of variables that you can reference in a parameter definition:

Type	Description
Special Variables	Special variables are build into Application Deployment. They represent information that is not determined until you specify a version and a target on the Deployment screen. A special variable pertains to one of three things: what is being deployed (the application, release, or version), where it is being deployed (the target or environment), or who is deploying it (the SA user). See "Special variables" on page 506.
Release Parameters	Release parameters are available to all components within a release. They are created and managed by the application owner and can be modified by anyone who has permission to edit the application. The value of a release parameter does not change unless it is explicitly modified. A release parameter has the same value within a specific version of the release. See "Release parameters" on page 507.
Global Parameters	Global parameters are available to all applications. They are created and managed by the Application Deployment administrator. The value specification of a global parameter does not change unless it is explicitly modified by the administrator. The actual value of a global parameter can vary depending on the environment and target. See "Global parameters" on page 508.

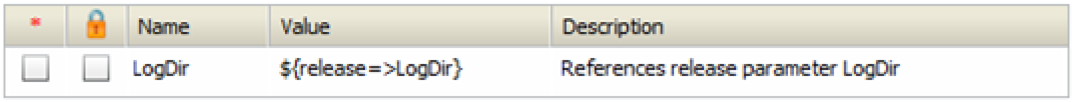
All three types of variables can be referenced by using the Select Special Variable dialog:



This dialog can be accessed at different points in the Application Deployment process (see "Accessing the select special variable dialog" on page 513).

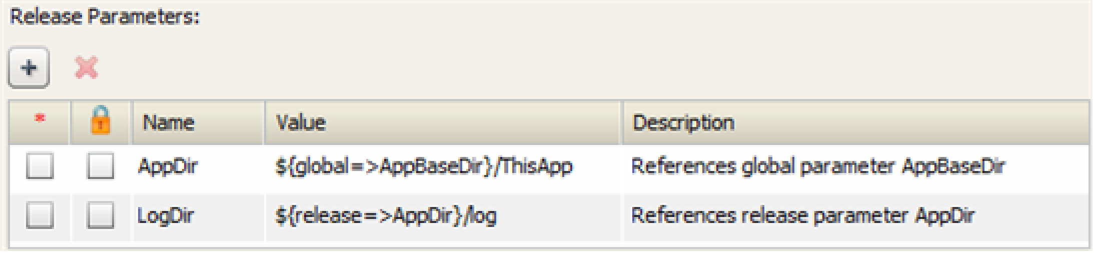
At deployment time, Application Deployment resolves the value of each component parameter by recursively following any references to special variables, release parameters, and global parameters.

For example, say that a component parameter named `LogDir` is defined as follows:



*	🔒	Name	Value	Description
☐	☐	LogDir	${release=>LogDir}	References release parameter LogDir

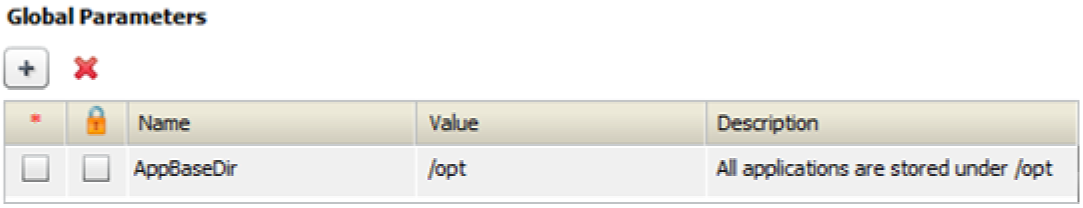
The release parameters, `AppDir` and `LogDir`, are defined as follows:



Release Parameters:

*	🔒	Name	Value	Description
☐	☐	AppDir	${global=>AppBaseDir}/ThisApp	References global parameter AppBaseDir
☐	☐	LogDir	${release=>AppDir}/log	References release parameter AppDir

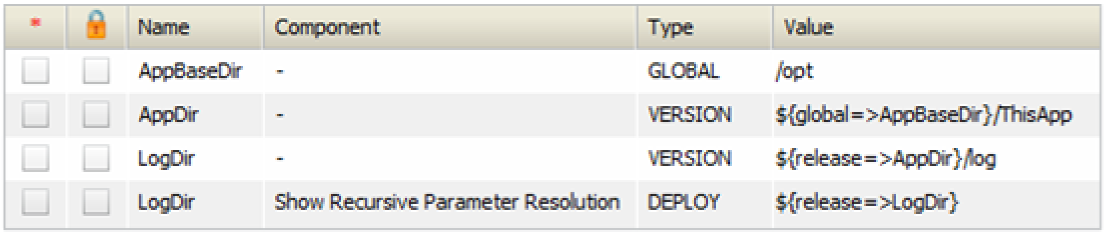
The global parameter `AppBaseDir` is defined to be `/opt`:



Global Parameters

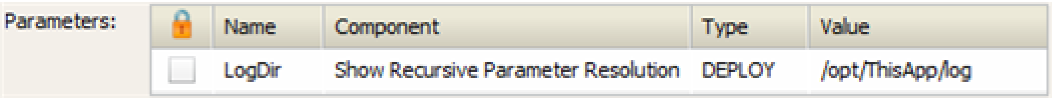
*	🔒	Name	Value	Description
☐	☐	AppBaseDir	/opt	All applications are stored under /opt

The default parameters look like this in the version:



*	🔒	Name	Component	Type	Value
☐	☐	AppBaseDir	-	GLOBAL	/opt
☐	☐	AppDir	-	VERSION	${global=>AppBaseDir}/ThisApp
☐	☐	LogDir	-	VERSION	${release=>AppDir}/log
☐	☐	LogDir	Show Recursive Parameter Resolution	DEPLOY	${release=>LogDir}

Assuming that there are no target or environment overrides, the ultimate value of the component parameter `LogDir` will be `/opt/ThisApp/log` when the component is deployed:



Parameters:

🔒	Name	Component	Type	Value
□	LogDir	Show Recursive Parameter Resolution	DEPLOY	/opt/ThisApp/log

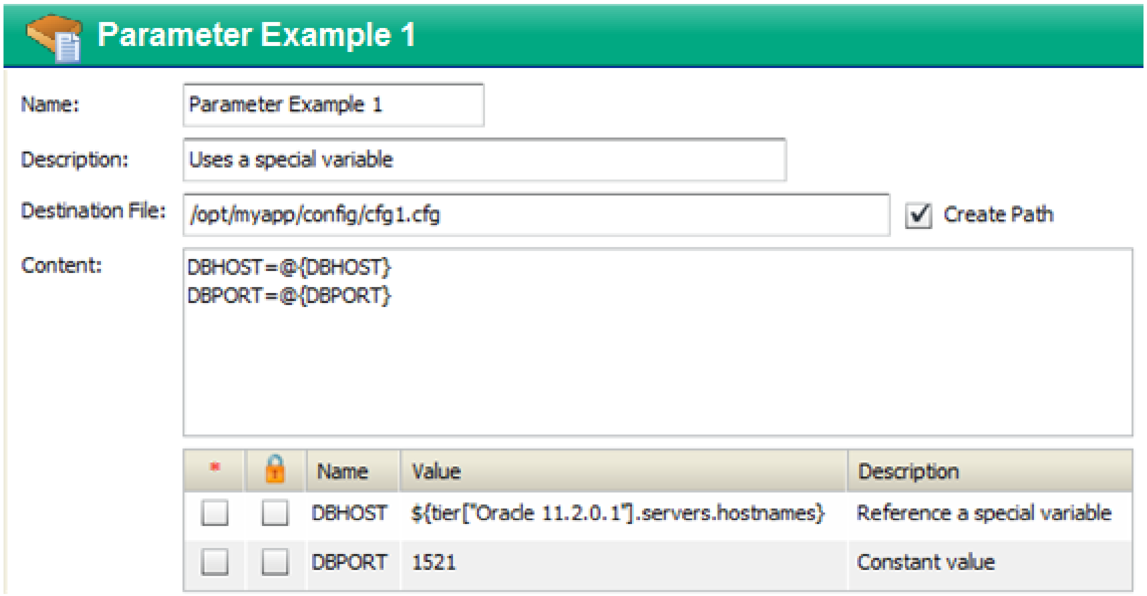
Special variables

Special variables represent information about the following types of items:

- The server where the component will be deployed
- The tier that contains that server
- The application to which the component belongs
- The release and version being deployed
- The environment where the targets reside
- The SA user who initiated the deployment

In the following example, the `DBHOST` parameter refers to a variable that represents a comma-separated list of the host names of the servers that are part of the Oracle 11.2.0.1 tier.

Parameter that references a special variable



Parameter Example 1

Name:	Parameter Example 1
Description:	Uses a special variable
Destination File:	/opt/myapp/config/cfg1.cfg ☑ Create Path

Content:

```
DBHOST=@{DBHOST}
DBPORT=@{DBPORT}
```

*	🔒	Name	Value	Description
□	□	DBHOST	${tier["Oracle 11.2.0.1"].servers.hostnames}	Reference a special variable
□	□	DBPORT	1521	Constant value

Although the values of some special variables may be known earlier, their values are not assigned (bound) until deployment time.

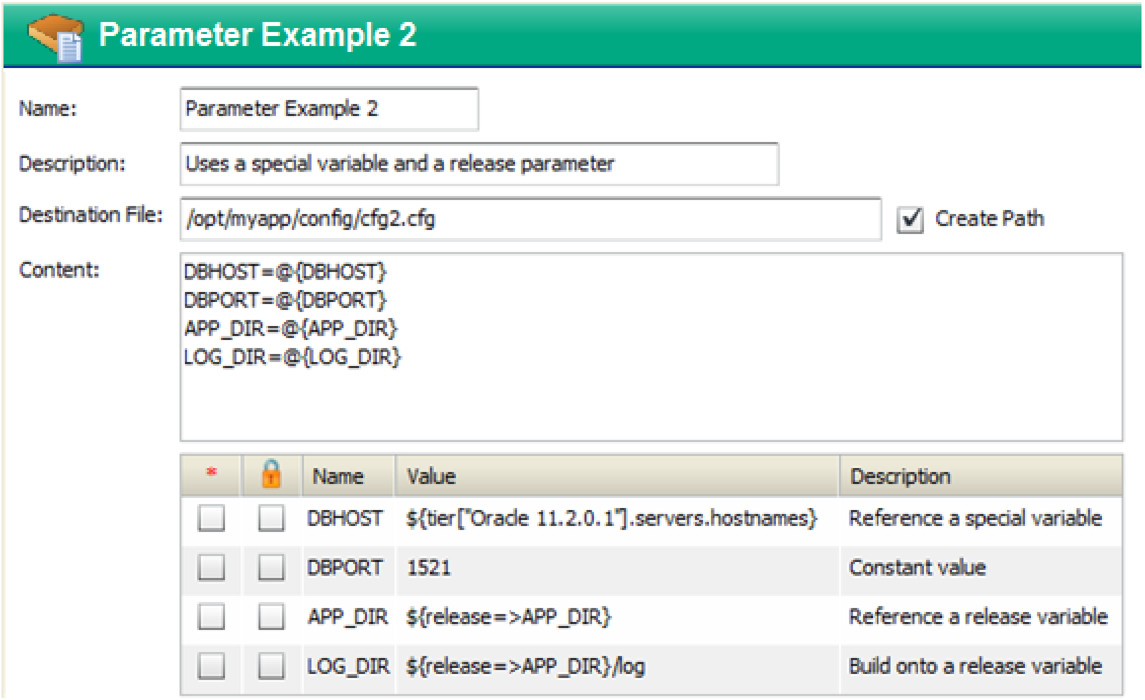
Note that “Server Name” special variables refer to the name of the server in SA. This name can be changed in SA without any change to the actual server host name. “Hostname” special variables refer to the actual host name of the server. The SA server name may or may not be the same as the host name.

For information about referencing a special variable in a parameter definition, see "Specifying the value of a parameter" on page 511.

Release parameters

Release parameters have values that are the same within a specific version of a release. They can be referenced by any component in the release. In the following example, the `APP_DIR` release parameter is used to represent the installation directory for the application:

Example of a release parameter



Parameter Example 2

Name: Parameter Example 2

Description: Uses a special variable and a release parameter

Destination File: /opt/myapp/config/cfg2.cfg ☑ Create Path

Content:

```
DBHOST=@{DBHOST}
DBPORT=@{DBPORT}
APP_DIR=@{APP_DIR}
LOG_DIR=@{LOG_DIR}
```

*	🔒	Name	Value	Description
□	□	DBHOST	${tier["Oracle 11.2.0.1"].servers.hostnames}	Reference a special variable
□	□	DBPORT	1521	Constant value
□	□	APP_DIR	${release=>APP_DIR}	Reference a release variable
□	□	LOG_DIR	${release=>APP_DIR}/log	Build onto a release variable

You can “build” the parameter value by adding information before or after the release parameter reference. In this example, the `LOG_DIR` parameter value adds the text “`/log`” to the value of the `APP_DIR` release parameter.

Release parameters can reference global parameters, special variables, and other release parameters (within the same release).

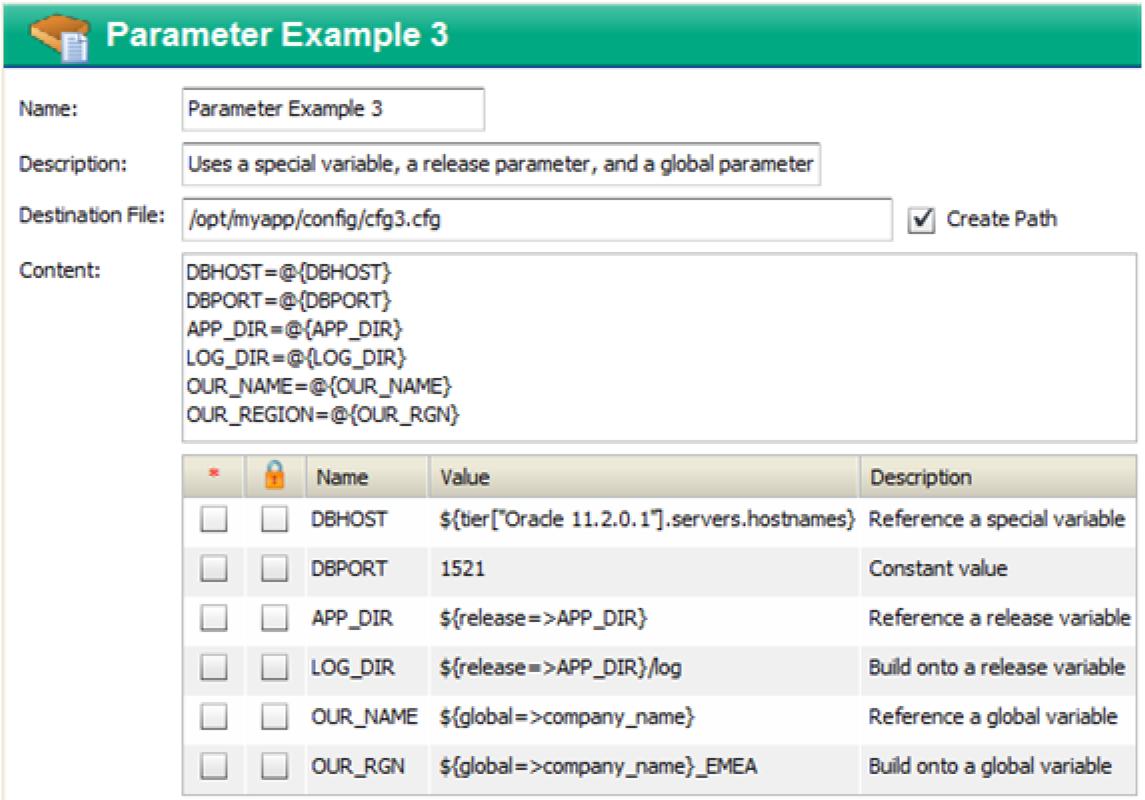
If you have permission to edit an application, you can define or modify release parameters for any release of that application (see "Defining a release parameter" on page 512).

When a new release is cloned from an existing release, the release parameters are also cloned.

Global parameters

Global parameters are the same across all applications. They can be referenced by any component in any application. They can also be referenced by release parameters. The following example uses the OUR_NAME global parameter to represent the name of the company:

Example with a global parameter



Parameter Example 3

Name:	Parameter Example 3	
Description:	Uses a special variable, a release parameter, and a global parameter	
Destination File:	/opt/myapp/config/cfg3.cfg	☑ Create Path

Content:

```
DBHOST=@{DBHOST}
DBPORT=@{DBPORT}
APP_DIR=@{APP_DIR}
LOG_DIR=@{LOG_DIR}
OUR_NAME=@{OUR_NAME}
OUR_REGION=@{OUR_RGN}
```

*	🔒	Name	Value	Description
☐	☐	DBHOST	${tier["Oracle 11.2.0.1"].servers.hostnames}	Reference a special variable
☐	☐	DBPORT	1521	Constant value
☐	☐	APP_DIR	${release=>APP_DIR}	Reference a release variable
☐	☐	LOG_DIR	${release=>APP_DIR}/log	Build onto a release variable
☐	☐	OUR_NAME	${global=>company_name}	Reference a global variable
☐	☐	OUR_RGN	${global=>company_name}_EMEA	Build onto a global variable

As you can with release parameters, you can build the parameter value by adding information before or after the global parameter reference. In this example, the OUR_RGN parameter value adds the text "_ EMEA" to the value of the company_name global parameter.

Global parameters can reference special variables and other global parameters.

Your Application Deployment administrator manages the global parameters (see "Administering application settings" on page 608 for more information).

Working with parameters and variables

The topics in this section provide “how to” information to help you do the following things:

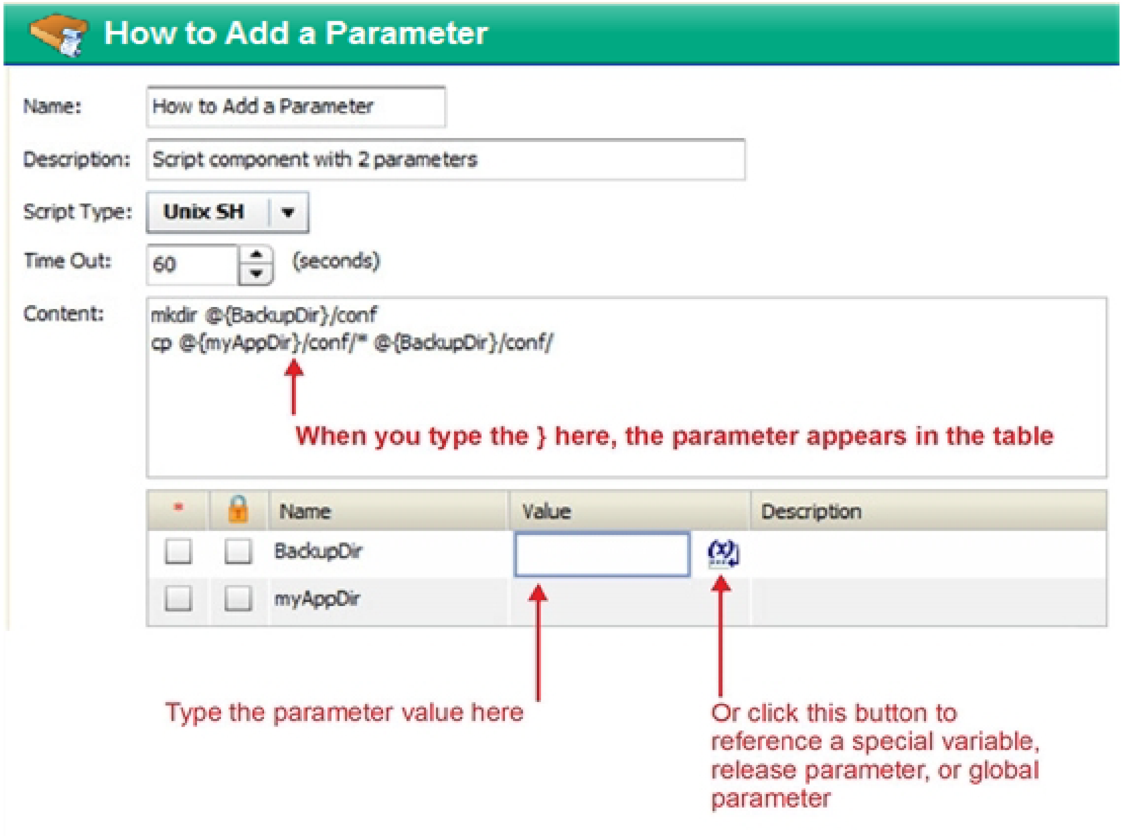
- "Specifying a component parameter" below
- "Specifying the value of a parameter" on page 511
- "Accessing the select special variable dialog" on page 513
- "Defining a release parameter" on page 512
- "Deleting a release parameter" on page 513

Specifying a component parameter

If you have permission to edit (or create) an application, you can add a parameter to the content of a Script or Configuration File component included in that application.

OO Flow and Application Configuration components can also use parameters. You can modify the value of the parameters for those component types, but you cannot add or delete the parameters themselves.

Example of a parameter reference in a script component



How to Add a Parameter

Name:	How to Add a Parameter
Description:	Script component with 2 parameters
Script Type:	Unix SH
Time Out:	60 (seconds)
Content:	

```
mkdir @{BackupDir}/conf
cp @{myAppDir}/conf/* @{BackupDir}/conf/
```

When you type the } here, the parameter appears in the table

*	🔒	Name	Value	Description
□	□	BackupDir		
□	□	myAppDir		

Type the parameter value here

Or click this button to reference a special variable, release parameter, or global parameter

To add a parameter to the content of a Script or Configuration File component:

1. In the Content box, type the name of the parameter by using the following notation:
   `@{parameter_name}`

   A parameter name may not contain the following characters: space, `@`, `\`, `{`, or `}`.
2. *Optional:* If the parameter is required to have a value when the component is deployed, select the box in the * (required) column.
3. Specify the parameter value. You can specify a constant value, reference a variable, or both (see "Specifying the value of a parameter" on the next page for instructions).
4. *Optional:* If you want the parameter to be encrypted, select the box in the 🔒 (encrypted) column.
5. Click the “Save Release” 💾 button on the tool bar to save your changes.

Specifying the value of a parameter

If you have permission to edit an application, you can change the default value of any of its component parameters. You can also change the value of its release parameters. You can specify a constant value (text or a number), or you can reference any of the items in the following table.

Variable reference formats

Variable Type	Format	Example
Special variable	`${special_variable_name}`	`${application.name}`
Release parameter	`${release=>parameter_name}`	`${release=>APP_DIR}`
Global parameter	`${global=>parameter_name}`	`${global=>APP_BASE_DIR}`

You can select a variable from a list of pertinent special variables, release parameters, and global parameters by using the Select Special Variable dialog.

You can combine text with variable references, as shown in these examples:

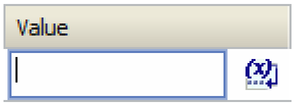
```
${release=>APP_DIR}/log
```

```
/var/${application.name}/my_data
```

```
${global=>TestBaseDir}/${application.name}
```

To specify the value of a parameter:

1. In the parameter table, click anywhere in the Value column for the parameter that you want to specify. A text box and a button appear in the Value column:

2. Follow these steps to select a special variable, release parameter, or global parameter reference from a list:

   a. Click the (x) button. The Select Special Variable dialog opens.

   b. Select the special variable, release parameter, or global parameter that you want to reference.

   c. Click **OK**. The variable reference now appears in the Value column.

3. *Optional:* Specify any additional text that you want to include in the parameter value.

4. Click **Save Release** on the tool bar to save your changes.

If you have Deploy permission for the application and Write permission for the pertinent environment, you can override the default parameter values at deployment time by following similar steps in the Parameter Editor (see "Customizing deployment parameters" on page 544).

Defining a release parameter

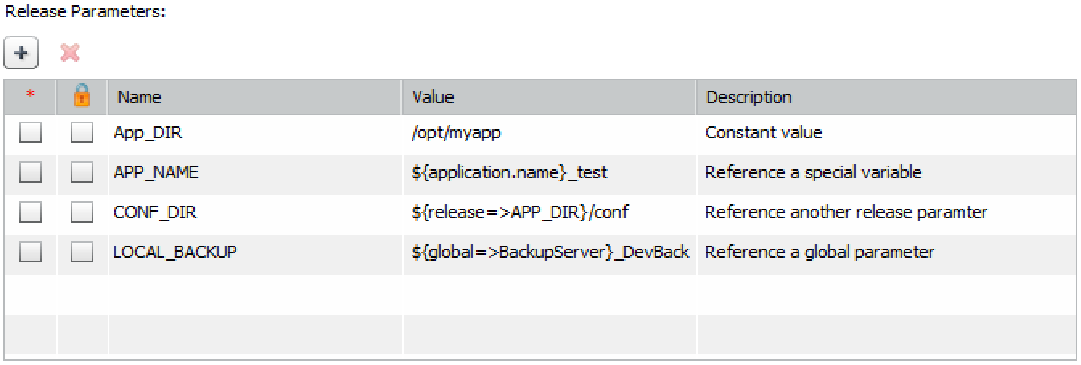
If you have permission to edit an application, you can define or modify its release parameters. These parameters are available to all components included in the release; they are not available outside the release.

To define or modify a release parameter:

1. On the Application screen, select the release that you want to work with.
2. On the tool bar, click **Edit release properties**. The Release Attributes dialog opens.

   You can also open this dialog from the Manage Applications tool (see "Managing applications" on page 460).
3. To add a new release parameter, follow these steps:
   a. Click **Add Release Parameter** button.
   b. Enter the Name of your new parameter. The name must be unique within the release parameters. The name cannot include any of these characters: @, \, {, or }.
   c. Click **OK**.
4. Click anywhere in the **Value** column for this parameter.
5. Specify the value of the parameter (for detailed instructions, see "Specifying the value of a parameter" on the previous page).

   You can reference special variables, other release parameters, and global parameters in the value specification. For example:



Release Parameters:

*	🔒	Name	Value	Description
□	□	App_DIR	/opt/myapp	Constant value
□	□	APP_NAME	${application.name}_test	Reference a special variable
□	□	CONF_DIR	${release=>APP_DIR}/conf	Reference another release paramter
□	□	LOCAL_BACKUP	${global=>BackupServer}_DevBack	Reference a global parameter

6. *Optional:* To encrypt the parameter value, select the box in the 🔒 (encrypted) column. Note that the * (required) column is not pertinent in this context.

7. Click **Save Release** on the tool bar to save your changes.

Deleting a release parameter

If you have permission to edit an application, you can delete any release parameters that are not referenced by any component parameters or other release parameters. You cannot delete a release parameter that is in use.

To delete a release parameter:

1. On the Application screen, select the release that you want to work with.
2. On the tool bar, click **Edit release properties**. The Release Attributes dialog opens.
3. In the **Release Parameters** table, select the parameter that you want to delete.
4. Click **Delete Parameter**. A confirmation dialog appears.
5. Click **Yes** to confirm the delete.
6. Click **Save Release** on the tool bar to save your changes.

Accessing the select special variable dialog

You can open the Select Special Variable Dialog from various places in the Application Deployment user interface.

How to open the select special variable dialog

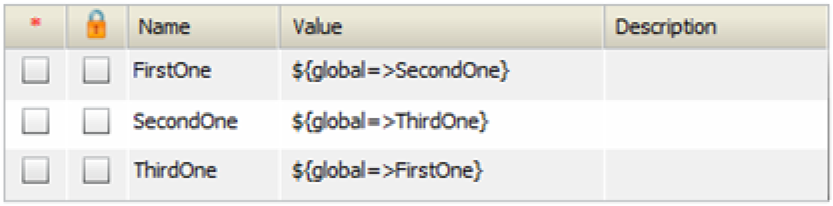
Location	Parameter Type	Instructions
Application screen	Component	In the component editor, click anywhere in the Value column for any component parameter.
Release	1. Click **Edit release properties**. 2. Click anywhere in the Value column for any release parameter.	1. Expand the Parameters section of the Deployment Options area. 2. Click **Edit** button in the upper left corner of the Parameters section. The Parameter Editor dialog opens. 3. In the right-most column, double-click the value of any parameter listed. The Parameter Values dialog opens. 4. Click the (x) button to the right of the Target Value or Environment Value box.
Deployment screen	Any	
Administration screen	Global	1. Go to the Application Settings page. 2. Click anywhere in the Value column for any release parameter.

Special considerations

Be aware of the following constraints and considerations as you specify parameter definitions.

Cycles

You cannot create circular references (cycles) in parameter definitions. The following specification, for example, will cause an error when you attempt to save the release:



*	🔒	Name	Value	Description
□	□	FirstOne	${global=>SecondOne}	
□	□	SecondOne	${global=>ThirdOne}	
□	□	ThirdOne	${global=>FirstOne}	

If a cycle is created after you save the application—for example, if the global parameters are later modified— you will not be able to create a deployment job for any new versions of this release. The Deployment screen will report an “infinite loop” error.

Self-referencing

Parameters are not allowed to reference themselves.

Deleting

If a component or release parameter in an existing version references a global parameter, that global parameter cannot be deleted.

If a component or release parameter in the current release references a release parameter, that release parameter cannot be deleted.

If a global parameter A references global parameter B, global parameter B cannot be deleted.

A deployment job will not be created if the version contains a reference to a global parameter that has been deleted.

The warning icon ⚠ will appear on the application tool bar if you save a release that includes a component that references a release parameter that has been deleted. Click the warning icon to determine which parameter has been deleted.

Naming

You cannot use the following characters in the names of component parameters, release parameters, or global parameters: @, \, {, or }.

You also cannot use spaces in the names of component parameters for Script or Configuration File components. Spaces are allowed in OO Flow component parameter names, release parameter names, and global parameter names.

Release parameter names must be unique within the release parameter list for a specific release.

Global parameter names must be unique within the global parameter list.

Release parameters may have the same names as component parameters or global parameters.

Global parameters may have the same names as component parameters or release parameters.

Importing and exporting

Global parameters and release parameters can be imported and exported (see "Importing and exporting data" on page 613).

Only those global parameters that are referenced by one or more release or component parameters will be imported when a partial import is performed.

Overriding

Environment or target value overrides created for a given release parameter will also be used for all releases that are cloned from the original release.

Target and environment overrides for global and release parameters follow the same order of precedence as component parameters. A target override supersedes an environment override, and an environment override supersedes the default value.

Targets

To view a simple example, see the "Quick start" on page 440.

Overview

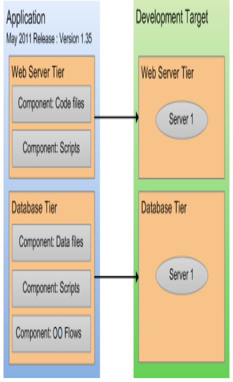
Applications are deployed to sets of managed servers. These sets of servers are called **targets**. Each target is generally suited to host one or more applications, depending on its structure.

Targets belong to specific **environments**, such as QA or Production. The environments and targets used in the Application Deployment user interface are “mirrored” as Device Groups in the main HPE Server Automation user interface.

A target is structured using **tiers**. A tier provides the functionality that a particular part of an application requires (typically middleware). Examples of tiers are web servers, application servers, and databases. When you deploy an application, all components in a given application tier are deployed to each server in the corresponding target tier.

An SA Software Policy may be associated with a tier to ensure that the proper middleware is either already in place on the tier or deployed along with applications managed by Application Deployment.

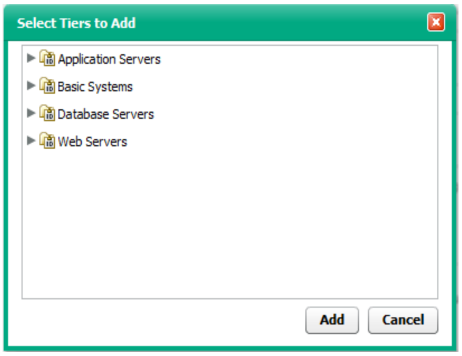
Target with two tiers



If you have a specific application in mind when you create a target, you can use that application's tiers to define the structure of the target. To do this, a deployable version of a specific release of the application must already exist. See "Deploying an application" on page 533 for more information.

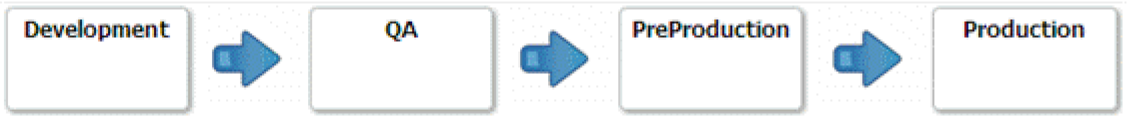
You can also create a new target and structure it yourself using existing tiers. The following tiers are provided "out of the box" with the SA:

Tiers provided



You can use these tiers to structure your targets, or you can define your own tiers (see "Administering Application Deployment" on page 593).

Each target is associated with a specific **environment**, such as QA or Production. Your environments map to your software development **lifecycle**.



You can construct a target in any environment that you have permission to edit (see "Setting permissions" on page 579).

By structuring your targets consistently across your environments, you can ensure that your test environments accurately reflect your Production environment.

The number of servers included in a target in one environment can be—and usually is—different than the number of items included in a target to which the same application is deployed in a different environment.

For example, in a simple QA environment, the same physical (or virtual) machine might be used to run both an application server and a database. In a Production environment, however, these responsibilities would be distributed among multiple machines. There might be a a dedicated database server and a cluster of application servers, for example.

Targets must have unique names within an Application Deployment environment.

Prerequisites

Before you can create or modify a target, the following conditions must be met:

- The tiers that this target requires are available.
- Any servers required to populate the tiers in this target are managed by SA.
- You have permission to create targets in this environment.
- You have permissions to view the servers that you want to add.

See the SA 10.50 Administration Guide for more information about SA permissions.

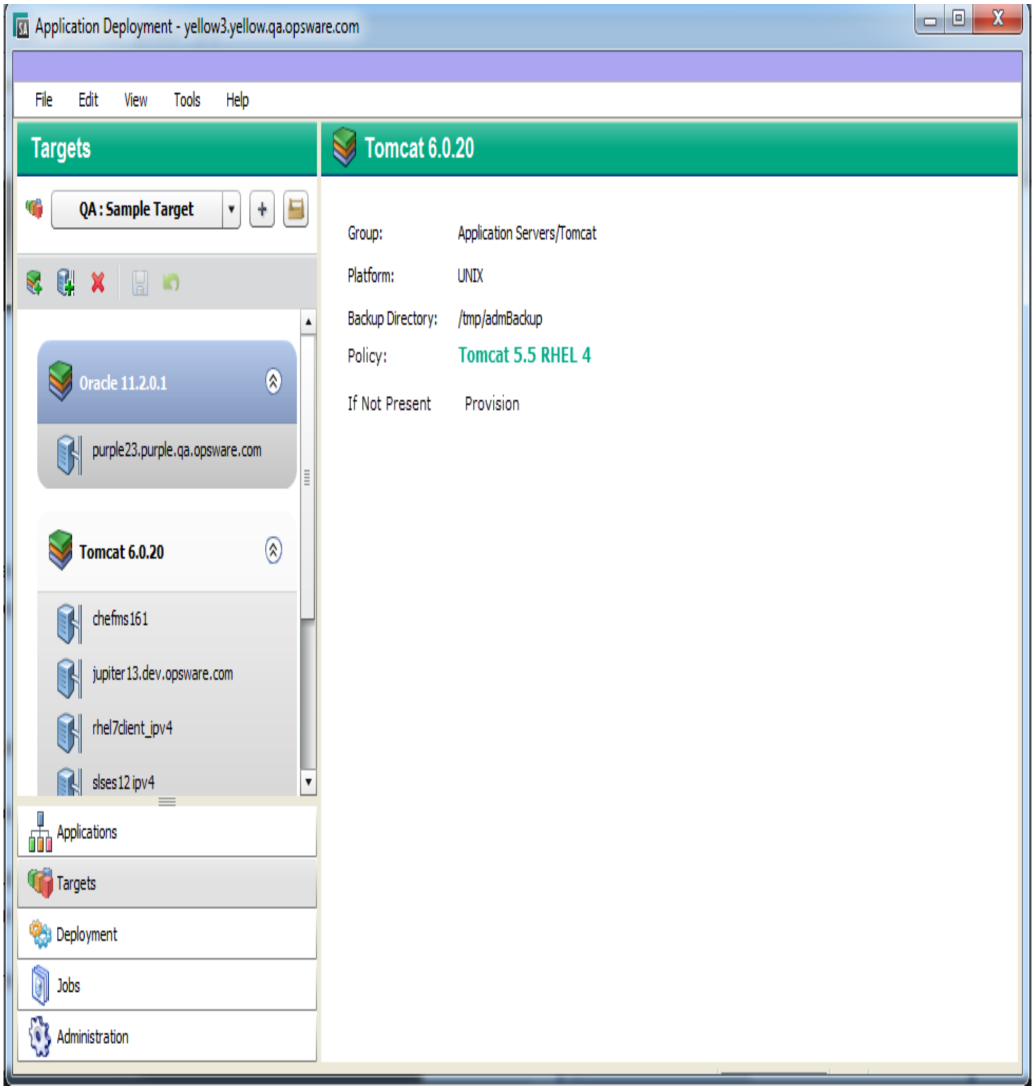
About the Targets screen

Before a target has been selected, the Targets screen looks like the image shown in "Quick start" on page 440. The following figure displays the target screen. After you select a target, it looks like .

Things to note:

- All existing targets are listed in the drop-down list in the upper left corner. Here, the Production: Web Store target has been selected.
- The left pane shows you the structure of the target.
- The right pane displays information about the item selected in the left pane. You can select a single server or a whole tier. When you select an item, the background color changes for that item. Here, the Oracle 11.2.0.1 tier is selected.
- Any blue text that appears in the right panel is a link to more detailed information. In this example, the link is to the policy that is associated with this tier. If you select a server in the left panel, you will see a link to more information about that server.

Targets screen showing an existing target



Application Deployment - yellow3.yellow.qa.opsware.com

Tomcat 6.0.20

Group: Application Servers/Tomcat

Platform: UNIX

Backup Directory: /tmp/admBackup

Policy: Tomcat 5.5 RHEL 4

If Not Present Provision

Working with targets

The following topics provide basic instructions for creating and maintaining targets.

- "Creating a new target" below
- "Managing targets" on page 522

For more detailed information, follow the links provided in each topic.

Creating a new target

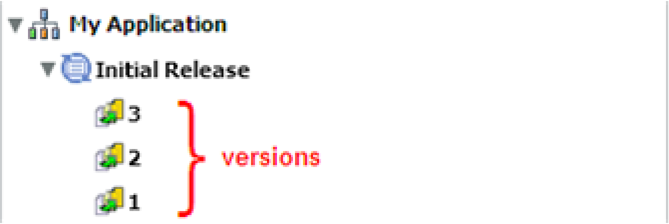
This topic provides basic instructions for creating a new target. It assumes that you have the SA permissions required to create Application Deployment targets and that you have permission to edit the environment in which the target will reside. See Prerequisites.

To create and configure a new target:

1. Go to the Targets screen (click **Targets** in the lower left corner).
2. Click **Create a New Target**. The Create New Target dialog opens.
3. Enter the target name. The name must be unique within the environment.
4. In the Location drop-down field, select the environment that will contain this target.

   There are two ways to populate your target with tiers: you can add them manually, or you can create them automatically.

   To automatically create the same set of tiers in your target that are used in an existing application, select “Use Application's Tiers” and select a version of the application.



Note that a specific version of a particular release of the application must have been previously created before it will appear in the Select Version drop-down list.

5. Click **OK**.
6. To add tiers to this target, follow these steps:
   a. Click **Add Tier**.
   b. Select the type of tier to add. You must create the same set (or a superset) of tiers that are used by your application.

      A group of standard tiers are provided with your software. If a tier that you need is not available, your Application Deployment administrator may need to add it. See "Administering tiers" on page 593.
   c. Click **Add**.
   d. Repeat these steps for each tier that your application requires.
7. For each tier that you added in step 6, follow these steps to specify the servers that will be included in this tier:
   a. Select the tier that you want to work with.
   b. Add a server to the tier.
8. Repeat step 7 to add servers to all remaining tiers.
9. Click **Save Target** to save your changes.

Adding a server

1. Click **Add Server**.

   The list of available servers is filtered by platform. Make sure that you select suitable servers for your tier. For example: You should select only Red Hat EL 5 servers if your components for that tier are intended for RedHat EL 5.
2. In the Add Servers to Target dialog, locate and select the managed server that you want to add. You can use the CTRL key to select multiple servers.

   If you are adding unprovisioned servers, see "Provisioning servers at deployment time" on page 529.
3. Click **OK**.

Managing targets

If you have the proper permissions (see "Overview" on page 579), you can use the Manage Targets tool to perform certain tasks. This tool enables you to do the following things:

- "Creating a new target group" on the next page
- "Deleting targets or target groups" on the next page
- "Renaming a target or target group" on page 524

To open the Manage Targets tool, click **Manage targets** on the Targets screen. If you do not have permission to view targets, this button is not available.

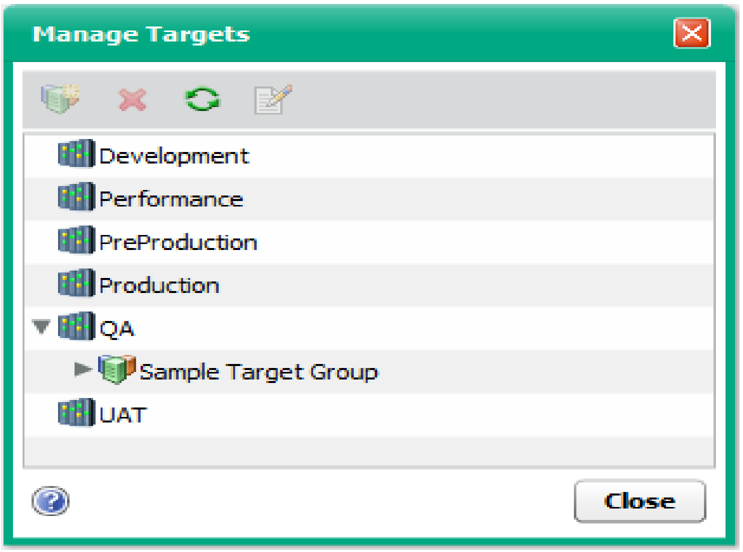
The Manage Targets tool contains a table that lists the following items:

Environments

Target groups (can be nested)

Targets

Manage Targets tool



Each item in the table that has “children” is preceded by an arrow. Click the ▶ arrow to expand an item; click the ▼ arrow to collapse it. To refresh your view and collapse all items, click the “Refresh” button ⟳.

To edit a specific item, select the row containing that item, and click the “Edit Properties” 📝 button on the tool bar—or simply double-click the item. You can only modify the name of a target or target group from the Manage Targets tool, however. To modify the structure or content of a target, return to the Targets screen.

Creating a new target group

A target group is a container that you can use to organize your targets in a hierarchy. If you have the proper permissions (see "Overview" on page 579), you can create target groups.

To create a target group:

1. Go to the Targets screen (click **Targets** in the lower left corner).
2. Open the Manage Targets tool (see "Managing targets" on the previous page).
3. Expand the environment in which you want to create the target group.
4. To create a target group at the top level, select the environment. To create a nested target group, select the existing group under which you want to create the new group.
5. Click **Create Target Group**.
6. Specify a name for your new group. The name must be unique within this container in the hierarchy.
7. Click **OK**. Your new group should now be visible in the table. The groups and targets are listed in alphabetical order at each level of the hierarchy.

You can drag and drop targets and target groups to different locations within the same environment. You cannot move them between environments, however.

Deleting targets or target groups

If you have the proper permissions (see "Overview" on page 579), you can use the Manage Targets tool to delete targets or empty target groups.

To delete an existing target or target group:

1. Open the Manage Targets tool (see "Managing targets" on page 522).
2. In the table, select the row containing the item that you want to delete. Note the following:
   - You can select multiple items by using the Ctrl or Shift key. You can only delete multiple items of the same type, however—for example, multiple targets or multiple target groups.
   - If you delete a target, any jobs that are scheduled but not running are also deleted.
   - You can only delete empty target groups.
3. Click **Delete**.

You will be asked if you really want to delete the item (or items). Click **Yes** to delete.

Renaming a target or target group

If you have the proper permissions (see "Overview" on page 579), you can change the name of a target or target group by using the Manage Targets tool.

To rename a target or group:

1. Open the Manage Targets tool (see "Managing targets" on page 522).
2. In the table, select the row containing the target or group that you want to rename.
3. Click **Edit Properties**.
4. Enter a new name for the target or group. Target names must be unique within the environment. Also see "Naming rules" on page 438.
5. Click **OK**.

Adding a tier to an existing target

This topic provides basic instructions for adding a tier to an existing target. It assumes that you have the SA permissions required to create Application Deployment targets and that you have permission to edit the environment in which the target resides. See Prerequisites for details.

To add a tier to an existing target:

1. Go to the Targets screen (click **Targets** in the lower left corner).
2. From the Select Target drop-down list, select the target where you want to add the tier.
3. Click **Add Tier**.

4. Select the type of tier to add. You must create the same set (or a superset) of tiers that are used by your application.

   A group of standard tiers are provided with your software. If a tier that you need is not available, your Application Deployment administrator may need to add it. See "Administering tiers" on page 593.
5. Click the **Add** button.
6. Add one or more servers to your new tier (see "Adding a server to an existing tier" below).
7. Click **Save Target** to save your changes.

Removing a tier from an existing target

This topic provides basic instructions for removing a tier from an existing target. It assumes that you have the SA permissions required to edit Application Deployment targets and that you have permission to edit the environment in which the target resides. See Prerequisites.

To remove a tier from an existing target:

1. Go to the Targets screen (click **Targets** in the lower left corner).
2. From the Select Target drop-down list, select the target from which you want to remove the tier.
3. Select the tier that you want to remove.
4. Click **Delete Selected Item**.
5. Click **Yes**.
6. Click **Save Target** to save your changes.

Adding a server to an existing tier

This topic provides basic instructions for adding a server to an existing tier. It assumes that you have the SA permissions required to edit Application Deployment targets and that you have permission to edit the environment in which the pertinent target resides. See Prerequisites.

To remove a tier from an existing target:

1. Go to the Targets screen (click **Targets** in the lower left corner).
2. From the Select Target drop-down list, select the target that you want to work with.

3. Select the tier where you want to add the server.
4. Click the **Save Target** to save your changes.

Removing a server from a tier

This topic provides basic instructions for removing a server from an existing tier. It assumes that you have the SA permissions required to edit Application Deployment targets and that you have permission to edit the environment in which the target resides. See Prerequisites.

To remove a server from an existing tier:

1. Go to the Targets screen (click **Targets** in the lower left corner).
2. From the Select Target drop-down list, select the target that you want to work with.
3. Select the server that you want to remove.
4. Click **Delete Selected Item**.
5. Click **Yes**.
6. Click **Save Target** to save your changes.

Note that this process only deletes the server from the target. It has no effect on the server itself.

Just-in-time targets

There are two methods for creating Application Deployment targets:

- You can create them prior to deployment by using the tools on the Targets screen (as described earlier in this section).
- You can create them when you deploy an application. This is called “just-in-time” target creation.

Just-in-time targets are useful when you want to delegate the allocation and/or selection of target resources. Just-in-time targets enable you to focus on the definition of the application and not on where it will be deployed. By using just-in-time targets and HPE Operations Orchestration (HPE OO) flows, you can externalize resource decisions and interface with 3rd-party allocation and scheduling tools.

Application Deployment populates just-in-time targets by using an HPE OO flow that you provide. In most cases, this flow will choose a set of existing managed servers and assign them to the new target.

In other cases, the flow will contain all the logic required to select a server, perform any necessary actions on that server, and deliver it to SA so that it can become a valid managed server.

After you create a just-in-time target, you can manage that target using the procedures described under "Managing targets" on page 522.

Prerequisites

Before you can create just-in-time targets, you must do the following things:

1. Configure SA to integrate with HPE OO.
2. Define an HPE OO flow that accepts the following parameters:

Parameter	Description
host	Host name or IP address of the SA core server.
username	Name of the SA user who will be creating the new target (this is needed to ensure that server allocation is constrained by this SA user’s permissions).
environment	ID of the environment that will cApplication Deployment contain the new target.
target	ID of the new target to which the flow will add servers.
tierCounts	A delimited string containing the tier and server count parameters. The format is `tier,count;tier,count...` The `tier` is the ID of the tier to which the flow will add the server. The `count` is the number of servers in the new (cloned or created from scratch) target.

3. Specify this HPE OO flow as the Just-In-Time Targets flow on the Administration screen in the Application Deployment user interface.

To create a just-in-time target, you must have Edit permission for the environment where the target will reside.

Creating a just-in-time target

To create a just-in-time target:

1. Select the **Deployment** button in the lower left corner.
2. Select the version of the application that you want to deploy. For more information, see "Deploying a specific version" on page 543.
3. Click **Advanced Target Selector** to the right of the Select Target drop-down list. The Advanced Target Selection dialog opens.
4. Select **Just-In-Time Target**.
5. Click **Continue**. The Create New Just-In-Time Target dialog opens. In this dialog, specify the following information:
   a. Name – type the name of the target.
   b. Location – select the environment or target group where the target will reside.
   c. Based On
      - Select Version to create the target with the same tiers that the version you are deploying contains.
      - This creates a true "on demand" target from scratch by choosing tiers based on the version.
      - Select Target to create the target by making a copy of an existing target.

        This creates a "cloned" target that uses another target's tier structure as a template. This is useful if you need to "flex up" resources at certain times—for example, if you have an online store, you might want to add more capacity during the holiday shopping season.
   d. If you selected Target in the previous step, select the existing target from the drop-down list.
   e. In the table, specify the number of servers that will be included in each tier (double-click the value in the New Size column).
6. Click **Create**. Application Deployment creates the new target and calls the Just-In-Time Targets flow to populate the target with servers. This flow is specified on the Administration page (see "Administering application settings" on page 608).

   If Application Deployment is unable to create the target, it will provide an error message.
7. Click **Close** to return to the Deployment screen.

   Your new target is now automatically selected. Click **Start Job** to initiate the deployment.

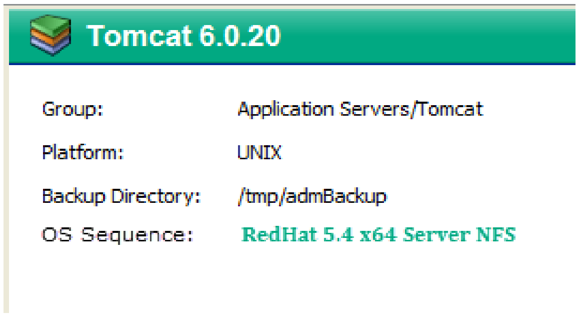
Provisioning servers at deployment time

Application Deployment enables you to install an operating system (OS) on an unprovisioned server at deployment time by using an OS Sequence.

An OS sequence defines what to install on an unprovisioned server, including OS build information from the installation profile and selected software policies.

There are three basic steps required to provision a server at deployment time:

1. Your Application Deployment administrator associates an OS Sequence (and any required middleware) with a specific tier:

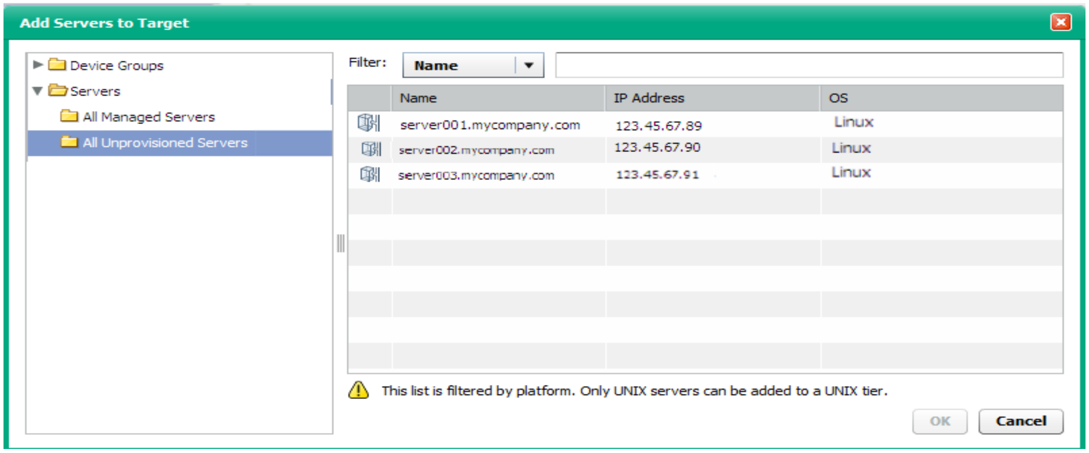
   

   Tomcat 6.0.20

Group:	Application Servers/Tomcat
Platform:	UNIX
Backup Directory:	/tmp/admBackup
OS Sequence:	RedHat 5.4 x64 Server NFS

   For more information, see "Administering tiers" on page 593.

2. When you define your target, you add one or more servers from the unprovisioned servers pool to this tier in your target:



Name	IP Address	OS
server001.mycompany.com	123.45.67.89	Linux
server002.mycompany.com	123.45.67.90	Linux
server003.mycompany.com	123.45.67.91	Linux

This list is filtered by platform. Only UNIX servers can be added to a UNIX tier.

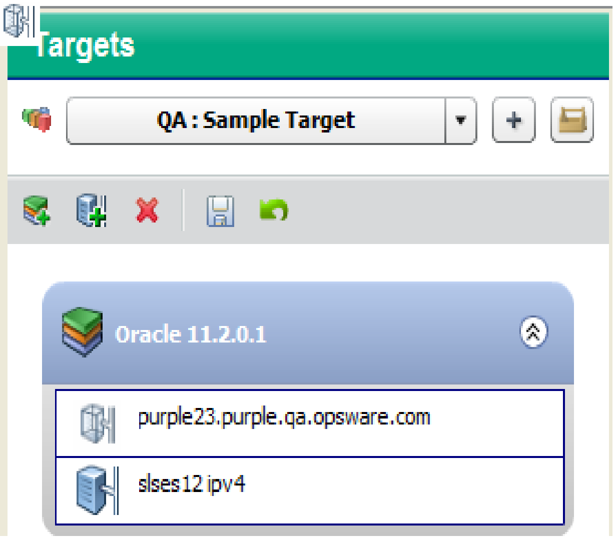
3.

   In order to see the All Unprovisioned Servers folder, you must have Server Pool permission on your SA Core. Your SA administrator must grant this permission to you (see the "Permissions Reference" section in the SA 10.50 Administration Guide).

You can also find Unprovisioned servers by browsing the Device Groups folder.

- Unprovisioned servers that boot as SA mini-agents have a hollow icon.
- Managed servers have a solid icon.

  These icons appear in the Add Servers to Target dialog (as shown above) and in the left pane on both the Targets screen and the Deployment screen:



4. When you deploy a version of an application to your target, Application Deployment recognizes that one or more servers in the target are unprovisioned. At staging time, it deploys the specified OS to those servers according to the information specified in the OS Sequence.

Prerequisites

Before Application Deployment can deploy an operating system to an unprovisioned server, the following conditions must be met:

- The server record must be in the SA Available lifecycle and the Unreachable SA state. This can be done using non–OGFS PXE or non–OGFS CD boot

  See the “Agent Management” section in the SA 10.50 User Guide for more information.

- To view the All Unprovisioned Servers folder inside the Application Deployment user interface, you must have Server Pool permission on your SA Core.
- The Library on your SA Core must contain an OS Sequence that can be used to appropriately provision the server.

  The person who is responsible for associating OS Sequences with tiers (presumably your Application Deployment administrator) must have the following SA permissions:
  - Manage OS Sequence feature permission
  - Folder Read permission for the folder where OS Sequences are saved

  These permission must come from a single SA user group.

  You do not need these permissions to provision a server at deployment time, however.
- The tier to which this server belongs must have an OS Sequence associated with it.

Servers that boot with OGFS PXE or OGFS CD boot cannot be provisioned using this process (in this version of SA). See OS Provisioning in the SA Online Help for more information.

How provisioning works

When you deploy a version of an application, Application Deployment performs the following steps at staging time:

1. Runs the Pre-Deployment Flow (if any) associated with the environment.
2. Runs the OS Sequences (if any) associated with the tiers in the specified target. This installs and configures the specified OS on any unprovisioned servers in the target.
3. Attaches and remediates any Policies associated with the tiers in the specified target.

Application Deployment then deploys the version.

You can include unprovisioned servers in both traditionally defined and just-in-time targets.

Please be aware of the following considerations:

Platform match

You can only add unprovisioned servers with the correct build agent type to a tier. For example, only servers with Linux build agents can be added to a tier with a Linux OS Sequence. Similarly, only Solaris build agents can install Solaris.

Presence of an OS sequence

You can only add an unprovisioned server to a tier that has an OS Sequence. If you add an unprovisioned server to a tier, and the OS Sequence is later removed from that tier, Application Deployment will fail to create a deployment job, and an error message will be displayed.

Conflicts

Sometimes an OS Sequence conflict can arise. If a single server is included in two different tiers, and the tiers have different OS Sequences, Application Deployment does not know which OS to install on the server. In this case, Application Deployment will fail to create a deployment job, and an error message will be displayed.

Rollback and undeploy

Rollback and Undeploy do not apply to OS Sequences. If you roll back or undeploy a job, any server that was unprovisioned at the time that you deployed that job and then provisioned using an OS Sequence will remain a managed server. You must explicitly deactivate a server before you can return it to the unprovisioned pool.

No re-provisioning

If you deploy to a target with a tier that contains both provisioned and unprovisioned servers, and an OS Sequence is associated with that tier, the OS Sequence will only be applied to the unprovisioned servers. In other words, you cannot re-provision a server using this process.

Server renaming

Depending upon how a server is configured, its name may change when it is provisioned using an OS Sequence. Typically, for example, bare metal machines will have generic names, and the OS Sequence may assign a more meaningful name. Similarly, when you deactivate a previously managed server and return it to the unprovisioned pool, its name may change when it is once again provisioned using an OS Sequence.

Although the Jobs log will reflect any renaming that occurs, the Application Deployment Jobs screen may not show the correct server name at first. You may need to close and re-open the Jobs screen to see the new name.

Provisioning failure

If an OS Sequence installation fails for any reason, the deployment job will fail, and the server’s state will be set to Provision Failed. If this happens, you must first deactivate (or delete) the server and then re-install the SA agent on that server. See the "Agent Management" section in the SA 10.50 User Guide for instructions.

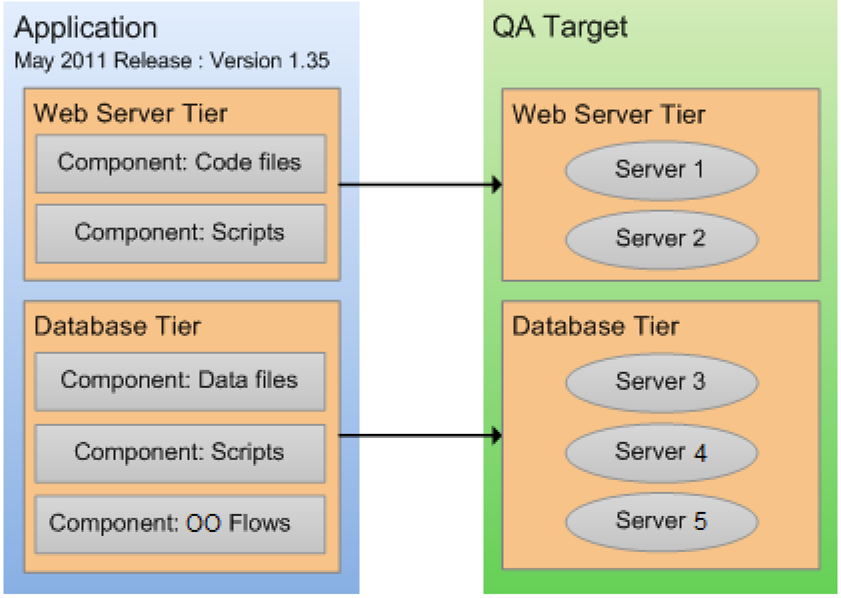
Deploying an application

To view a simple example, see the "Quick start" on page 440.

Overview

In Application Deployment, you deploy a specific **version** of your application to a **target**. The components of the application are deployed to all servers in the pertinent tier of the specified target. The components are deployed in the order shown in the Deployment Order view (see "Components" on page 468). Unless rolling deployment is configured, all servers must finish before the next component is deployed.

Deploying an application to a target



The deployment of a component has four phases:

1. Any necessary files are copied from the SA Core to a temporary location on the target server. This is called **staging**.For Software Policy components—and Code and Package components that use the Policy deploy method—the policy is attached during staging.
2. Application Deployment waits until the specified **cut over** time.
3. If Rollback is enabled in the component, pertinent files and directories on the target server are backed up in the location that was specified when the tier was defined (see "Administering Application Deployment" on page 593).
4. The action required to deploy the component is performed. The nature of this action depends on the component type:

- For a Code component:
  - If the deploy method is Package, the files are extracted from the staged ZIP file and placed in the specified location on the target server.
  - If the deploy method is Policy, the policy attached during staging is remediated.
- For a Script component, the specified script is executed on the target server.
- For a Configuration File component, the file is created on the target server.
- For a Software Policy component, the policy is attached during staging is remediated.
- For a Package component, one of two things happens:
  - If the deploy method is Package, the packages are installed on the target server.
  - If the deploy method is Policy, the policy attached during staging is remediated.

- For an Application Configuration component, the Application Configuration is deployed to the target server.
- For an OO Flow component, the flow is run to completion on the HPE Operations Orchestration server.
- For a Windows Registry component, the pertinent registry keys or values are added or deleted.

Application Deployment creates a deployment job that will perform these steps for each component included in the selected version for each server in the specified target (or targets). You can view the progress and status of the job on the Jobs screen.

If any component fails to deploy properly or the Deployment job is cancelled, the deployment will automatically **roll back** from the point of failure. For each component for which rollback is enabled, the rollback action specified in the component is performed, and any staging files stored in the temporary directory are removed.

You can also manually roll back a deployment from the Jobs screen. A rollback returns the server to the state that it was in prior to that deployment.

An alternative to rolling back is **undeploying**. The undeploy process uninstalls the version and removes any trace of it on the server. It is a more aggressive process than a rollback. It does not attempt to return the server to its previous state.

When you deploy a version, you can instruct Application Deployment to automatically undeploy the previous version. In this case, two jobs are created: the undeploy job for the previous version, and the deploy job for the current version.

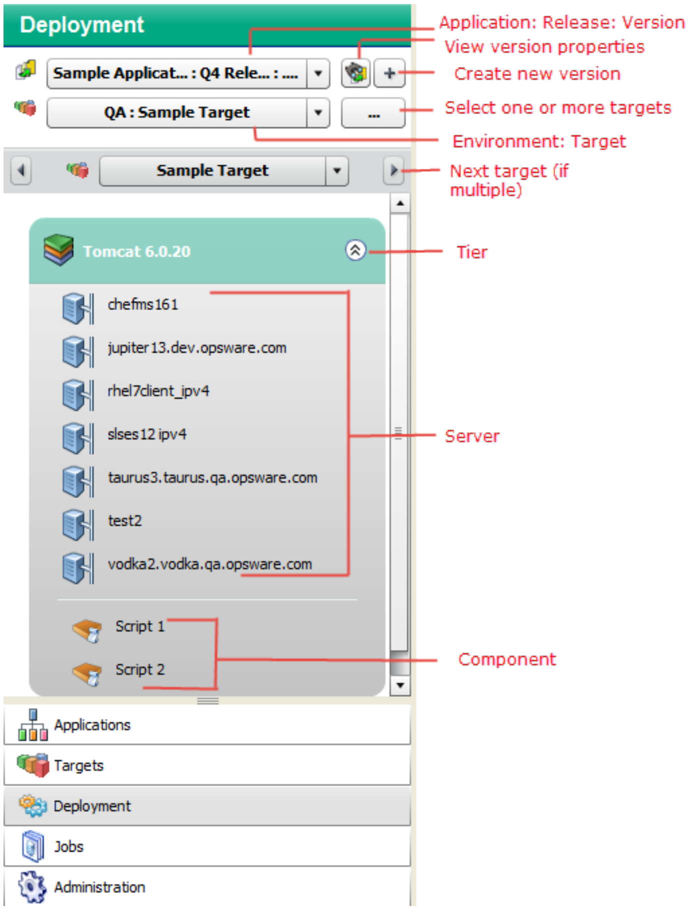
Application Deployment supports **rolling deployment**. This means that you can deploy the components in a particular tier to a subset of the target servers in that tier while keeping the application running (live) on the rest of the servers in the tier.

- You can customize the parameters used by the components at deployment time. This is useful if you want to override the default values for specific target or environment or simply use different values for a particular deployment.
- As of SA version 9.01, SA supports “just-in-time” targets—targets that are created at deployment time using an HPE OO flow that you provide. See "Targets" on page 516 for more information.

Deployment screen

The Deployment screen enables you to deploy a specific version of a release to one or more targets. The left panel of the deployment screen represents the structural view.

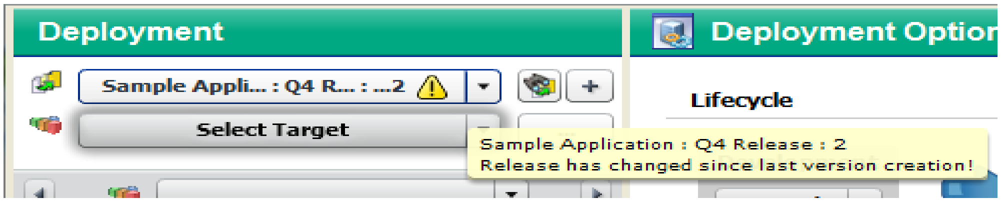
Deployment screen left panel



In this example, a single target has been selected (Advantage123), and each tier has exactly one server. You can select multiple targets for a deployment, but all targets must be in the same environment.

The “Warning” sign ⚠ is visible here, because the Green Checking 1.0 release has changed since the RC 3 version was created. When this icon appears, hover over it to see more information.

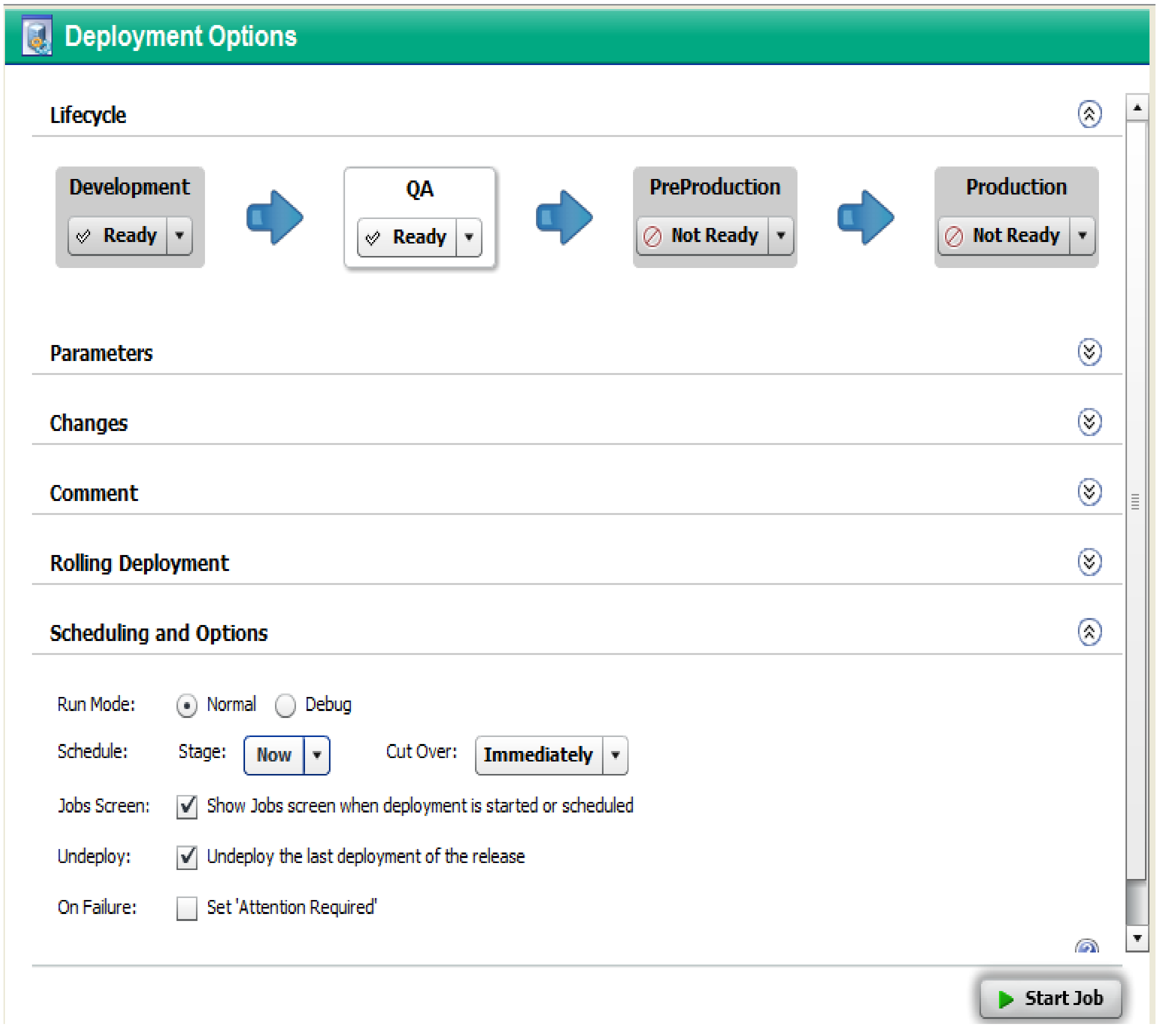
Tool tip associated with Warning sign



Sample Application : Q4 Release : 2
Release has changed since last version creation!

The right panel of the Deployment screen shows you the content view—the settings that are specific to this deployment job.

Deployment screen right panel



Development: ✓ Ready
QA: ✓ Ready
PreProduction: ⊘ Not Ready
Production: ⊘ Not Ready

Run Mode: ● Normal ○ Debug

Schedule: Stage: Now Cut Over: Immediately

Jobs Screen: ☑ Show Jobs screen when deployment is started or scheduled

Undeploy: ☑ Undeploy the last deployment of the release

On Failure: □ Set 'Attention Required'

The following sections in the right panel contain information about the deployment job that will be created when you click the **Start Job** button:

- Lifecycle
- "Parameters" on the next page

- "Changes" on the next page
- "Comment" on the next page
- "Rolling deployment" on page 540
- "Scheduling and options" on page 540

To expand a section, click the ⌄ icon (or double-click the blue bar).

Lifecycle

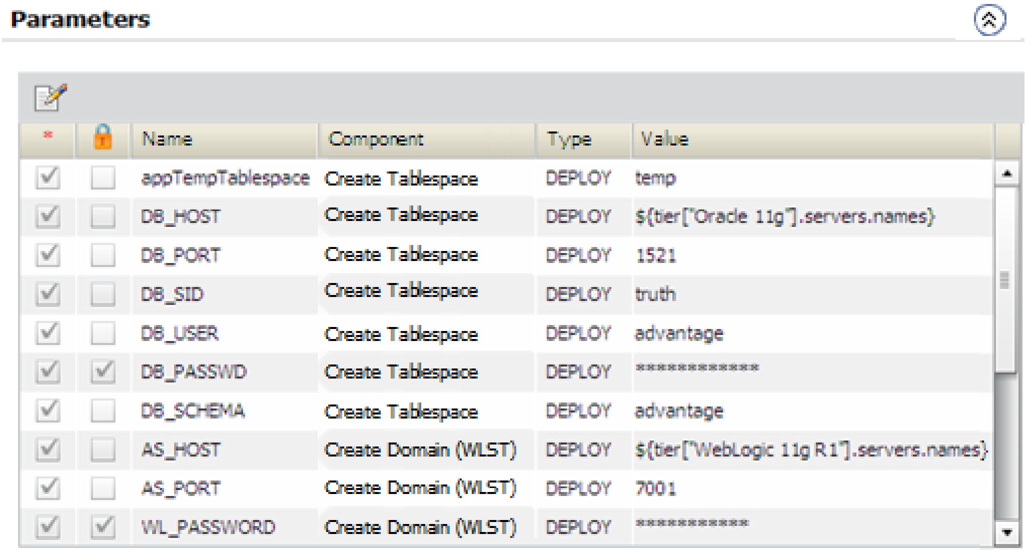
The Lifecycle section at the top of the panel shows you the status of all the environments included in this lifecycle. The environment to which you are deploying is highlighted.

You can only deploy to any environments that are *not* marked Not Ready. Your Application Deployment administrator or the environment owner determines which environments are available. Anyone with Deploy permission for this environment can update the status of the environment at deployment time.

Parameters

The Parameters section enables you to view and customize the values of any editable Parameters used by the application’s components. At deployment time, Application Deployment checks to make sure that all required parameters have values—if they do not, the Deployment job will not start.

Parameters information on the Deployment Job screen



Parameters

*	🔒	Name	Component	Type	Value
☑	□	appTempTablespace	Create Tablespace	DEPLOY	temp
☑	□	DB_HOST	Create Tablespace	DEPLOY	${tier["Oracle 11g"].servers.names}
☑	□	DB_PORT	Create Tablespace	DEPLOY	1521
☑	□	DB_SID	Create Tablespace	DEPLOY	truth
☑	□	DB_USER	Create Tablespace	DEPLOY	advantage
☑	☑	DB_PASSWD	Create Tablespace	DEPLOY	***************
☑	□	DB_SCHEMA	Create Tablespace	DEPLOY	advantage
☑	□	AS_HOST	Create Domain (WLST)	DEPLOY	${tier["WebLogic 11g R1"].servers.names}
☑	□	AS_PORT	Create Domain (WLST)	DEPLOY	7001
☑	☑	WL_PASSWORD	Create Domain (WLST)	DEPLOY	**************

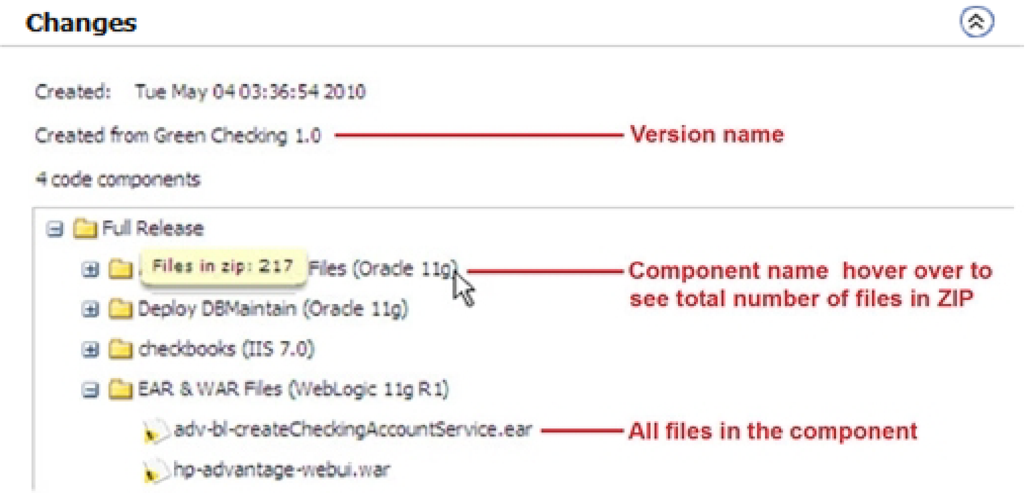
In this example, the parameters have the values that were specified in the components of the application—these are the “application default” values. The `DB_HOST` and `AS_HOST` parameters use “special variables” and take their values from the names of the servers in their respective tiers. Special variables are placeholders for data that cannot be known until deployment time (when destination servers are chosen).

You can override the application values at deployment time for a specific target or an entire environment (see "Customizing deployment parameters" on page 544).

Changes

The Changes section shows you how the Code component files associated with this release have changed. For a full release, all files are listed (as shown here). For a delta release, any files that have been added or modified since the “baseline” version of the release was deployed are listed.

Code component changes listed on the Deployment Job screen



Changes

Created: Tue May 04 03:36:54 2010

Created from Green Checking 1.0

4 code components

- Full Release
  - Files in zip: 217 Files (Oracle 11g)
  - Deploy DBMaintain (Oracle 11g)
  - checkbooks (IIS 7.0)
  - EAR & WAR Files (WebLogic 11g R1)
    - adv-bl-createCheckingAccountService.ear
    - hp-advantage-webui.war

Version name

Component name hover over to see total number of files in ZIP

All files in the component

Comment

The Comment section provides a place where you can enter a text comment. This will be displayed in the reports and in the Job Logs.

Rolling deployment

The Rolling Deployment section enables you to decide how many servers in each tier to take offline for deployment purposes at the same time. The Number of Servers per Group setting controls this. By default, the setting is the number of servers in the tier—by default, all servers will be deployed at once.

For example, an application should be written such that servers are removed from a load-balanced pool while they are updated and then added back in. This ensures that the majority of serves remain in service while the rolling batch is updated.

If a tier has nine servers, you might choose to deploy each component in the corresponding application tier to three servers at a time before proceeding to the next group of three. In this case, you would set the Number of Servers per Group to three.

If Rolling Deployment is used, groups of components in the same tier will be deployed to the specified number of servers in the tier. If Rolling Deployment is not used, a component will not be deployed until the previous component has been successfully deployed to all servers in the tier.

Scheduling and options

The Scheduling and Options section enables you to specify how and when the deployment is performed, including options for debugging (see "Managing Application Deployment jobs" on page 555).

Deployment Job Options

Option	Purpose
Run Mode	Determines whether the deployment job should run normally or in debug mode. If you select “Debug” here, the Debugger window will open when you click the **Start Job** button.
Stage	Determines when Application Deployment should copy any files required for the deployment from the SA core to a temporary directory on the target servers. This option is automatically set to “Now” when the Run Mode is “Debug.”
Cut Over	Determines when the components should actually be deployed on the targets. At the specified Cut Over time, a backup of pertinent files and directories is performed (provided that Rollback is enabled), and then the components are deployed. Be careful when scheduling the Cut Over time. For example, if Staging requires 20 minutes to complete, you should schedule the Cut Over at least 30 minutes later. Application Deployment will not allow you to schedule the Cut Over time

Deployment Job Options, continued

Option	Purpose
	ahead of the Staging time. If you specify “Immediately,” the backup and deployment steps will begin as soon as the files are staged. This options is automatically set to “Immediately” when the Run Mode is “Debug.”
Jobs Screen	Determines whether the Jobs screen is displayed after you click the **Start Job** button. If you select the “Show job status when deployment starts” box, the Jobs screen will open. A filter is applied such that only your Deployment job is visible. You can watch the Job Logs as each step in the deployment is performed. If you do not select this option, the Deployment screen remains displayed. This option is automatically selected when the Run Mode is “Debug.”
Undeploy	Determines whether the previous deployment of this version is undeployed before the current deployment happens. If the “Undeploy the last deployment of this release” box is not selected, Application Deployment makes no attempt to undeploy the previous deployment. This option is not available when the Run Mode is “Debug.”
On Failure	Determines whether the Deployment job should be placed in Attention Required status prior to rolling back if the job fails. When a Deployment job fails, one of two things happens: If this option is *not* selected, Application Deployment sets the job status to Failed and performs an automatic rollback. If this option is selected, the job is placed in a suspended state (Attention Required) so that you can check the status of your managed servers to troubleshoot the failure. After you have collected enough information about the failure, you can resume the job. Application Deployment will then set the job status to Failed and initiate the automatic rollback. This option is automatically selected when the Run Mode is “Debug.”

Enabling the Job button

The **Start Job** button becomes enabled after both a version and a target have been selected. Click the button to schedule or start the Deployment job according to the options that you have specified.

If any required parameters do not have values, or if the version and target have a mismatch (for example: components are for Red Hat Linux but target servers are running SUSE Linux), the Deployment job will not start.

Deploying a version

When you deploy an application, you deploy a specific version of that application. A version is associated with a particular release. There are two steps required to deploy a version:

- "Creating a new version" below (or using an existing version)
- "Deploying a specific version" on the next page

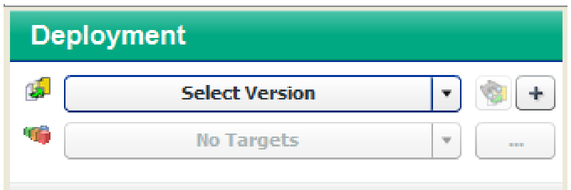
You can deploy any application that you have permission to deploy provided that you have Deploy permission for the environment (see "Overview" on page 579).

Creating a new version

Provided that you have permission to deploy an application (see "Overview" on page 579), you can create a new version of a specific release of that application.

To create a version of your application:

1. Select the **Deployment** tab in the lower left corner.
2. In the Select Version drop-down field, navigate to your application.



3. Under the pertinent release of your application, click the **Create New Version** link.
4. In the Create New Version window, enter a Version name (or number).

   The version name is simply a string that you specify. If the previous version of this release contains a number, however, Application Deployment will automatically increment it. If the previous version was “1,” for example, “2” will appear in the Version field. If the previous version was “V 1.2.1,” the string “V 1.2.2” will appear in the Version field.

Regardless of what Application Deployment puts in the Version field, you can specify any version name that you want. The version name must be unique within a release, however.

5. *Optional:* In the Description field, enter any additional information that you want to specify for this version.
6. *Optional:* If you want to see a list of the code components that have been updated since the previous version was created, select **Show Code Component Changes**.
7. Click **Create**. This creates a new version of your application by gathering all the files that are components of this application.

   This starts a job that creates the version. The progress bar in the lower left shows you the status and progress of the job. If the job fails, go to the Jobs screen to find out why.
8. Select **Close**. A new version of your application is now ready for deployment.

Deploying a specific version

You deploy a **version** of your application to one or more **targets**. Application Deployment translates component definitions (from your application definition) into **deployment steps** that will be executed on servers specified in the selected targets. The deployment steps do the following things:

Copy the application files to the target servers.

Execute scripts that you provided with your application.

Apply items from the SA Library that you have specified.

Execute OO flows referenced by the application.

To deploy a version:

1. Select the **Deployment** button in the lower left corner.
2. From the Select Version drop-down list, navigate to and select the version of the application that you want to deploy. Select an existing version, or create a new one (see "Creating a new version" on the previous page).

   When you hover over an existing version in the list, its tiers are displayed to the right of the list. If a release is marked with an asterisk (*), this means that this release of the application has been modified since the last version was created.

3. From the Select Target drop-down list, select the target where you want this version of the application to be deployed.

   If you do not see your target in the list, make sure that the following conditions are true:

   - The tiers of the target match (or are a superset of) the application tiers.
   - You have Deploy permission for this environment.
   - Each tier contains at least one server.
   - The environment is *not* marked Not Ready in the lifecycle.
   - The lifecycle for the release containing your version must include your environment and target.

   For additional target options, click **Advanced Target Selector** to the right of the Select Target drop-down list:

   - You can specify **multiple targets** if you want to deploy this version to more than one target. Only those targets that meet all of the requirements listed above can be selected.
   - If your SA Core and Application Deployment are configured to support them, you can specify a **just-in-time** target to create a new target for this deployment. This option is available only when the following conditions are true:
     - Your SA core is configured to run HPE Operations Orchestration (HPE OO) flows.
     - A Just-In-Time Targets Flow has been specified in the Application Settings area on the Administration screen.

   For more information, see "Just-in-time targets" on page 526.

4. *Optional:* Review the deployment job settings in the right panel. You can modify some of these settings (see "Deployment screen" on page 536). To expand a section, click the ⌄ icon.

5. Click **Start Job**. This launches a Deployment job that will copy your application components to the target server and run any necessary code, scripts, and flows.

You can see the status of the deployment on the Jobs screen.

Customizing deployment parameters

Parameters are place-holders used to deliver information that can vary depending on the environment or target where an application is deployed. The application default values for a parameter can be different between versions. You might, for example, use different database users in different versions.

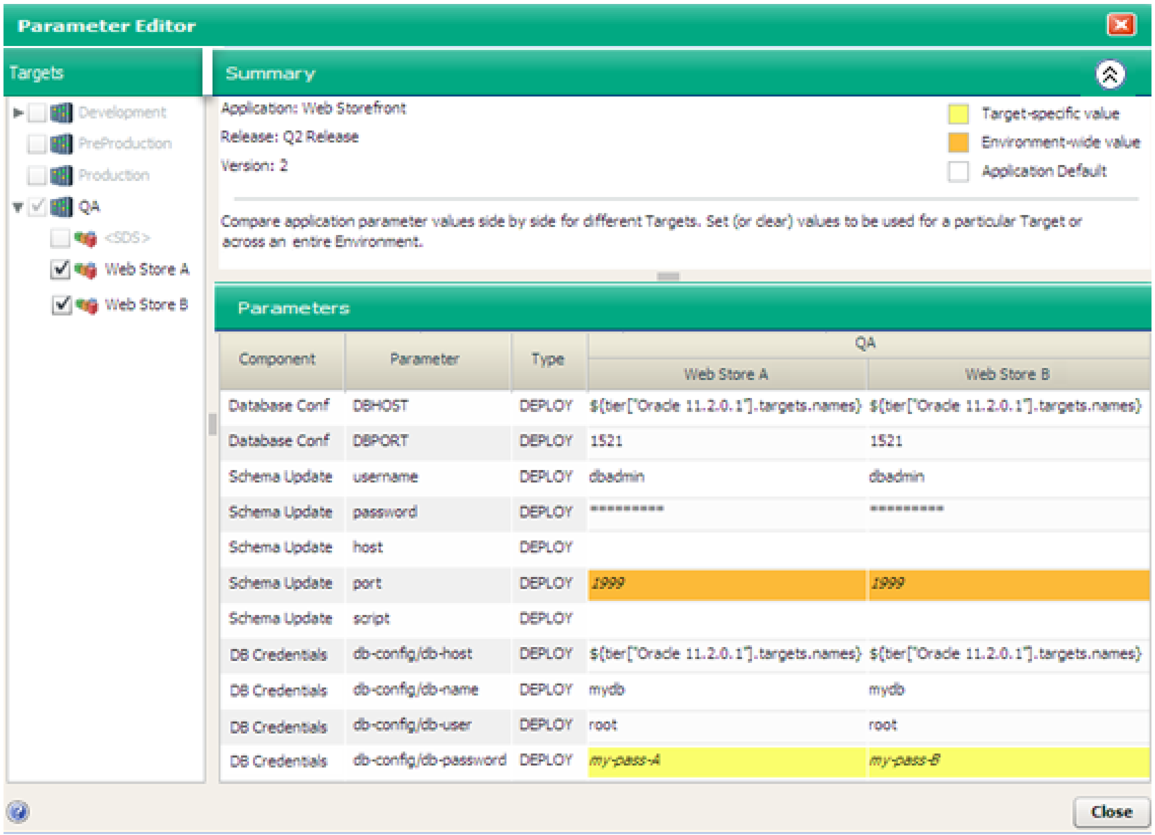
Parameters are initially defined in the components that specify the functionality of an application (see "Components" on page 468). Their values can be set three ways:

- In the application, itself, when the component is specified.
- In the environment where the application will be deployed (by the person who is deploying the application).
- For a specific target at deployment time. Application Deployment remembers target-specific values, so you do not have to set them for every deployment.

Parameters are specific to a component within an application. As long as the name of the parameter doesn't change, and the component is cloned from the original component at deployment time, the value will be re-used for each new version.

You can use the Parameter Editor to establish the values of an application's parameters at deployment time. If you are deploying to multiple targets, you can customize these values for each target. The following example includes two targets in the QA environment: Web Store A and Web Store B.

Parameter editor example



Application: Web Storefront
Release: Q2 Release
Version: 2

🟨 Target-specific value
🟧 Environment-wide value
□ Application Default

Compare application parameter values side by side for different Targets. Set (or clear) values to be used for a particular Target or across an entire Environment.

Component	Parameter	Type	QA	
			Web Store A	Web Store B
Database Conf	DBHOST	DEPLOY	${tier["Oracle 11.2.0.1"].targets.names}	${tier["Oracle 11.2.0.1"].targets.names}
Database Conf	DBPORT	DEPLOY	1521	1521
Schema Update	username	DEPLOY	dbadmin	dbadmin
Schema Update	password	DEPLOY	**********	**********
Schema Update	host	DEPLOY		
Schema Update	port	DEPLOY	*1999*	*1999*
Schema Update	script	DEPLOY		
DB Credentials	db-config/db-host	DEPLOY	${tier["Oracle 11.2.0.1"].targets.names}	${tier["Oracle 11.2.0.1"].targets.names}
DB Credentials	db-config/db-name	DEPLOY	mydb	mydb
DB Credentials	db-config/db-user	DEPLOY	root	root
DB Credentials	db-config/db-password	DEPLOY	*my-pass-A*	*my-pass-B*

Here, both targets are selected in the Targets list on the left, so the parameter values for both targets are shown in the Parameters table on the right. Only those targets that were selected for this deployment are available in the Targets list; other targets in the pertinent environment are grayed out.

The Type column indicates the source of each component parameter value (see "Parameters and special variables" on page 502):

Component Parameter Types

Type	Parameter Value Source
DEPLOY	Constant or special variable
VERSION	Release parameter
GLOBAL	Global parameter If a component parameter is encrypted, the Parameter Editor will not display global parameters referenced by that component parameter.

Most of the parameters for this deployment have the default values that were established when this release of the Web Storefront application was specified.

Two parameters (`DBHOST` in the Database Conf component and `db-config/db-host` in the DB Credentials component) use "special variables" and will get their values when this version is deployed. In this case, their values are the names of the servers in the Oracle 11.2.0.1 tier in the respective targets.

The `port` parameter in the Schema Update component gets its value from the QA environment, where it was configured by person who deployed the application.

The `db-config/db-password` parameter in the DB Credentials component has a different value for each target. In this example, the value is a constant, but it could also be a special variable.

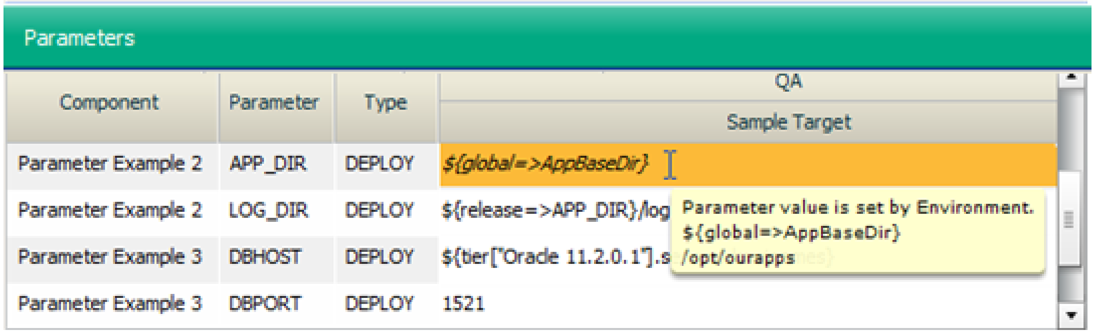
The order of precedence for parameters is as follows:

1. Target specific value (yellow background in the table)
2. Environment-wide value (orange background)
3. Application default (white background)

This means that an environment-wide value will override an application default. A target-specific value will override an environment-wide value or an application default.

When you hover the mouse over a parameter value, the resolved value of that parameter appears in a tool tip, which also indicates whether any overrides are in effect.

Parameter value tool tip example



Parameters

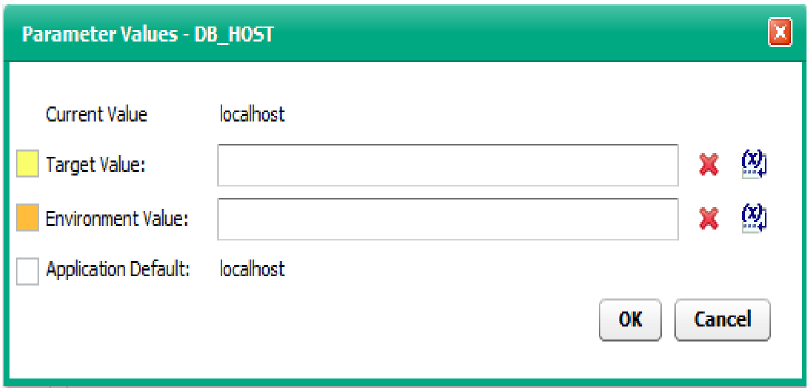
Component	Parameter	Type	QA
			Sample Target
Parameter Example 2	APP_DIR	DEPLOY	*${global=>AppBaseDir}*
Parameter Example 2	LOG_DIR	DEPLOY	${release=>APP_DIR}/log
Parameter Example 3	DBHOST	DEPLOY	${tier["Oracle 11.2.0.1"].s
Parameter Example 3	DBPORT	DEPLOY	1521

Parameter value is set by Environment.
${global=>AppBaseDir}
/opt/ourapps

In the following figure, the cursor is hovering over the value of the `APP_DIR` parameter. This parameter references the `AppBaseDir` global parameter, which resolves to `/opt/ourapps`. This is an environment override—as indicated in the tool tip and also by the orange background of this cell in the table.

To modify the value of a parameter for a specific target:

1. On the Deployment screen, specify a version and a target.
2. In the right panel, click the ⌄ icon to expand the Parameters section.
3. Either click **Edit**, or double-click any row in the table. The Parameter Editor opens.
4. In the Parameters table, double-click the value of a parameter. The Parameter Values dialog opens, as shown here:

   Parameter Values - DB_HOST

	Current Value	localhost
🟨	Target Value:	
🟧	Environment Value:	
□	Application Default:	localhost

   Note that encrypted values will be displayed as a series of asterisks (*).

   To specify a constant value, type it in the Target Value box.

   To reference a special variable, release parameter, or global parameter (see "Parameters and special variables" on page 502), follow these steps:

   a. Click the (x) icon.
   b. Select the item that you want to use from the list.
   c. Click **OK**.

   To erase any characters in the Target Value box—and reset the value to the next value in precedence order—click the “Delete” button: ✖. For example, if you delete the target value, and there is an environment value, the environment value is used. If there is no environment value, the application default value is used.

5. Click **OK** to close the Parameter Values dialog.

If you changed the parameter value for this target only, it will now have a yellow background in the Parameters table. If you applied the value to all targets in the environment, it will have an orange background.

Environment or target value overrides created for a given release parameter will also be used for all releases that are cloned from the original release.

Viewing the properties of a version

After you create a version, you can examine its properties by using the version viewer. You can open the version viewer from several places in the Application Deployment user interface.

Version viewer launch points

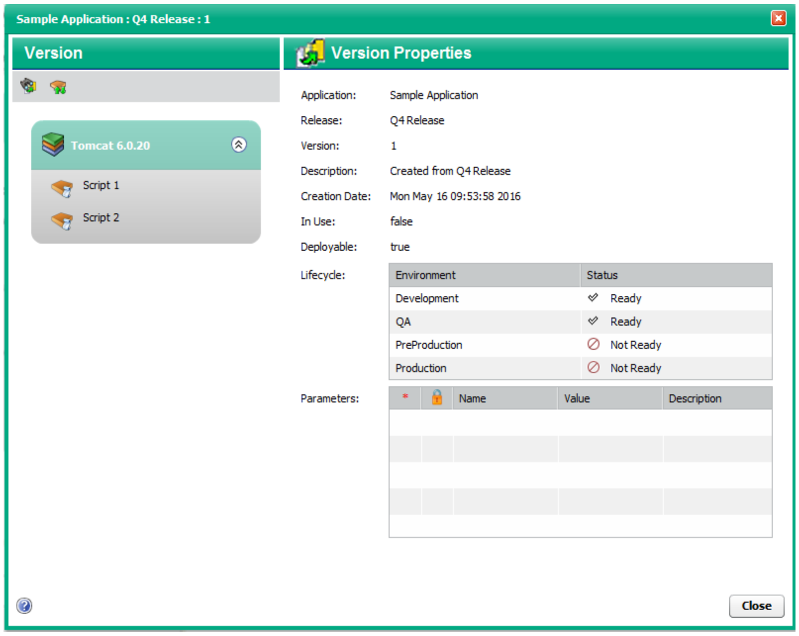
Location	How to open
Deployment screen	Click **View version** to the immediate right of the version selector.
Application screen	Open the Manage Applications tool, select a version in the table, and click **View version** on the tool bar.
Jobs screen	Select a Deployment job in the Job Logs table, and click **View version** on the tool bar. You can also open the version viewer by clicking the Application link in the Job details window.

The version viewer shows the version that is selected in the current context. It displays tier and component information in the left panel and version, tier, or component properties in the right panel. The information displayed in the version viewer is read-only.

When a version is created, the release information is immutable, but the tier information can change. The tier configuration specifies the middleware to be used for that tier. Because middleware can be updated, this allows old applications to use new middleware. The tier information displayed in the version viewer represents the current tier configuration.

In order to view the properties of a version, you must have View permission for the associated application (see "Application permissions" on page 589).

Version viewer – structural view



Application: Sample Application
Release: Q4 Release
Version: 1
Description: Created from Q4 Release
Creation Date: Mon May 16 09:53:58 2016
In Use: false
Deployable: true

Lifecycle:

Environment	Status
Development	✓ Ready
QA	✓ Ready
PreProduction	⊘ Not Ready
Production	⊘ Not Ready

Parameters:

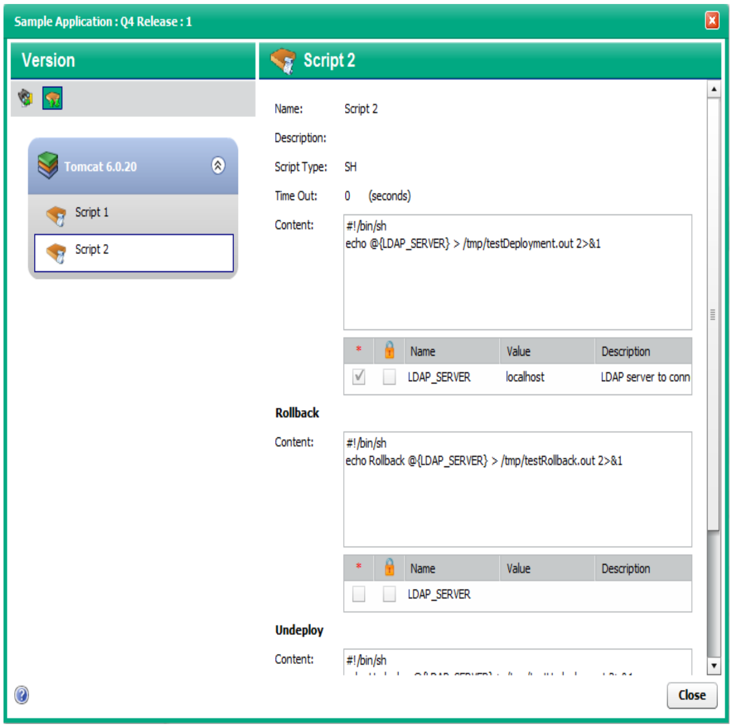
*	🔒	Name	Value	Description

You can view the information in the left panel in one of two ways:

- The **structural** view shows you the tiers included in the application and the components included in each tier.
- The **sequential** view shows you the order in which the components will be deployed.

In either view, select a tier or component in the left panel to display its properties in the right panel.

Version viewer – sequential view



Name: Script 2

Description:

Script Type: SH

Time Out: 0 (seconds)

Content:

```
#!/bin/sh
echo @{LDAP_SERVER} > /tmp/testDeployment.out 2>&1
```

*	🔒	Name	Value	Description
☑	□	LDAP_SERVER	localhost	LDAP server to conn

Rollback

Content:

```
#!/bin/sh
echo Rollback @{LDAP_SERVER} > /tmp/testRollback.out 2>&1
```

*	🔒	Name	Value	Description
□	□	LDAP_SERVER		

Undeploy

Content:

```
#!/bin/sh
```

You can further control the information that is displayed by using the tool bar buttons.

Version Viewer tool bar buttons

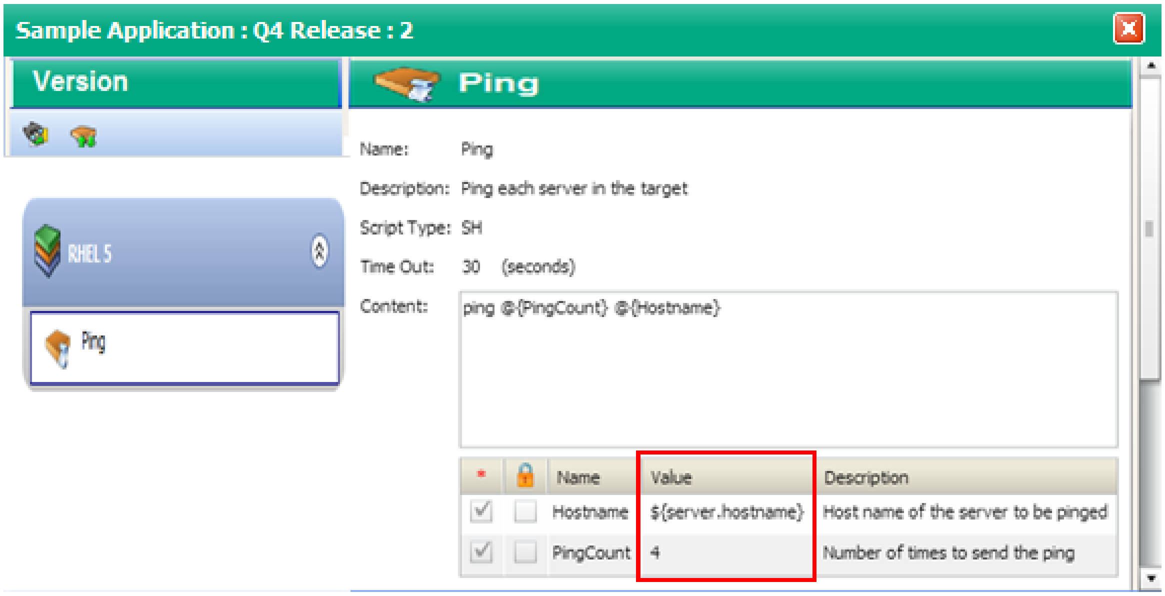
Button	Purpose
	In the left panel, list the components in the order they were (or will be) deployed.
	Change from the component order (sequential) view to the tier (structural) view in the left panel.
	Display the version properties in the right panel.

Note the following:

- When you view an application on the Application screen, you see the *names* of the Backup, Rollback, and Undeploy scripts if they are enabled. In the version viewer, however, you see the *contents* of each enabled script at the time that the version was created. The contents of the scripts are copied into the version so that subsequent script changes do not affect existing versions.
- The parameter values shown in the version viewer are the *default* values for that component. If this version has already been deployed, the values shown in the version viewer might not match the deployment-specific values that are shown in the Job or Step detail windows for that deployment job (see "Jobs screen " on page 555).

In the following simplified example, an application includes a script component called Ping, which simply pings each server in the target. In the version viewer, you see the default values of the parameters:

Version viewer parameter list



Name: Ping

Description: Ping each server in the target

Script Type: SH

Time Out: 30 (seconds)

Content:

```
ping @{PingCount} @{Hostname}
```

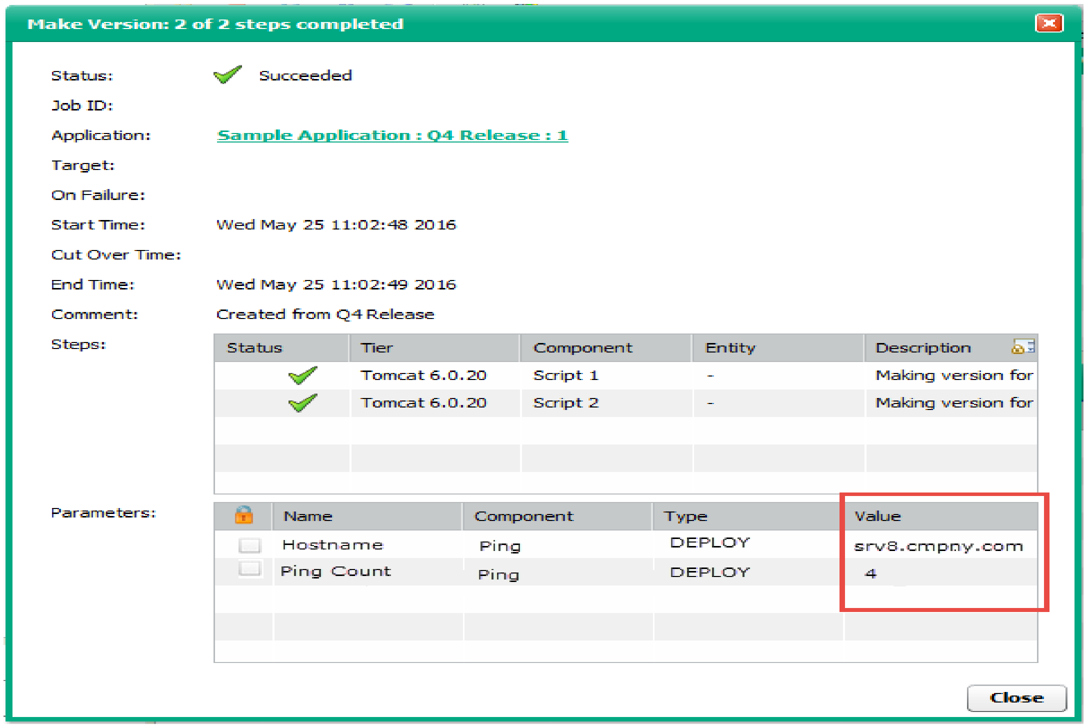
*	🔒	Name	Value	Description
☑	□	Hostname	${server.hostname}	Host name of the server to be pinged
☑	□	PingCount	4	Number of times to send the ping

When you deploy a version of this application to a server, however, the Hostname parameter contains the actual host name of that server.

You can see this substitution in the Job details window for this deployment job. In this example, the special variable `${server.hostname}` was assigned the value `srv8.compny.com` at deployment time.

Note that the PingCount parameter is a constant (not a special variable). In this case, neither the target nor the environment overrides the default value at deployment time. Therefore, the parameter has the same value (4) in both views.

Job details window parameter list



Status: ✓ Succeeded
Job ID:
Application: Sample Application : Q4 Release : 1
Target:
On Failure:
Start Time: Wed May 25 11:02:48 2016
Cut Over Time:
End Time: Wed May 25 11:02:49 2016
Comment: Created from Q4 Release
Steps:

Status	Tier	Component	Entity	Description
✓	Tomcat 6.0.20	Script 1	-	Making version for
✓	Tomcat 6.0.20	Script 2	-	Making version for

Parameters:

	Name	Component	Type	Value
□	Hostname	Ping	DEPLOY	srv8.cmpny.com
□	Ping Count	Ping	DEPLOY	4

Rolling back a deployment

If a failure occurs during the deployment of a version, that deployment is automatically rolled back from the point of failure. From the Jobs screen, you can manually roll back any version that you have deployed.

Regardless of whether a rollback operation is initiated automatically or manually, the process and outcome is the same.

A rollback operation removes the components in the reverse order in which they were deployed and restores the pertinent files, directories, or policies to their previous state. The rollback instructions are specified when the component is defined. If rollback is disabled for a component, that component is not rolled back.

Rollback behavior by component type

Component Type	Rollback Behavior
Code	Execute the OS-specific rollback and restore scripts maintained by your Application Deployment administrator (see "Administering Application Deployment" on page 593).
Script	Execute the rollback script, if one is specified in the component.

Rollback behavior by component type, continued

Component Type	Rollback Behavior
Configuration File	Execute the OS-specific restore script maintained by your Application Deployment administrator.
Application Configuration	Remove the Application Configuration from the server, and restore the original file (if one was saved).
Software Policy	Detach the Policy from the server, restore the related existing policy (if any), and remediate.
Package	If the deploy method is Policy, detach and remediate the policy containing the packages, and attach and remediate the related policy (if any). If the deploy method is Package, remove the packages, and install any related packages (if any).
OO Flow	Initiate the specified OO Rollback Flow (if one was specified).
DMA Flow	Execute the specified DMA Flow (if one was specified).
Windows Registry	Execute the Windows-specific rollback script maintained by your Application Deployment administrator.

A rollback operation returns the server to the state that it was in prior to the deployment.

The standard Rollback scripts maintain only one backup version for each release—they remove any existing backup files in order to conserve disk space on the target server. This means that if you roll back a version that is older than the most recently deployed version, Application Deployment will not be able to restore the code files. If you attempt to roll back a previous version, Application Deployment displays a warning message. You can then either confirm or cancel the rollback.

To manually rollback a deployment:

1. Go to the Jobs screen (click Jobs in the lower left corner).
2. In the Jobs Log table, select the row corresponding to the deployment that you want to roll back.
3. Click **Rollback**.
   The Rollback Job dialog opens.
4. Click **Yes**. A rollback job is created and started.

It is possible that there is nothing to roll back, because the only components that executed did not have rollback enabled. If this is the case, you will see a warning message.

Undeploying a deployment

When you deploy a version, you can request that the previous version be undeployed—if a previous version was, indeed, deployed. From the Jobs screen, you can manually undeploy the most recent version that you have deployed. Your Application Deployment administrator can undeploy any version.

Regardless of whether an undeploy operation is initiated automatically or manually, the process and outcome is the same.

An undeploy operation removes the components in the reverse order in which they were deployed. The undeploy instructions are specified when the component is defined. If undeploy is disabled for a component, that component is not removed.

Undeploy behavior by component type

Component type	Undeploy behavior
Code	Remove the files from the default install location.
Script	Execute the undeploy script, if one was specified in the component.
Configuration File	Execute the OS-specific undeploy script maintained by your Application Deployment administrator.
Application Configuration	Remove the Application Configuration.
Software Policy	Detach the Policy from the server, and remediate.
Package	If the deploy method is Policy, detach and remediate the policy containing the packages. If the deploy method is Package, remove the packages.
OO Flow	Initiate the specified OO Undeploy Flow (if one was specified).
DMA Flow	Execute the specified DMA Flow (if one was specified).
Windows Registry	Execute the Windows-specific undeploy script maintained by your Application Deployment administrator.

An undeploy operation removes any trace of the version. It does not attempt to return the server to a specific state.

To manually undeploy a version:

1. Go to the Jobs screen (click Jobs in the lower left corner).
2. In the Jobs Log table, select the row corresponding to the deployment that you want to undeploy.
3. Click **Undeploy**. The Undeploy Job dialog opens.
4. Click **Yes**. An undeploy job is created and started.

Managing Application Deployment jobs

To view a simple example, see the "Quick start" on page 440.

Overview

Application Deployment creates and runs the following types of jobs:

- A Make Version job is created each time that you create a new version.
- A Deployment job is created each time that you deploy a version (one job per target).
- A Rollback job is created whenever a deployment is rolled back—either automatically due to a deployment failure or manually (one job per target).
- An Undeployment job is created whenever a version is undeployed—either automatically before a newer version is deployed or manually (one job per target).

A job consists of a series of steps. Multiple steps are required to deploy, roll back, or undeploy each component. The number of steps depends on the type of components used, the number of components used, and the number of servers.

Jobs screen

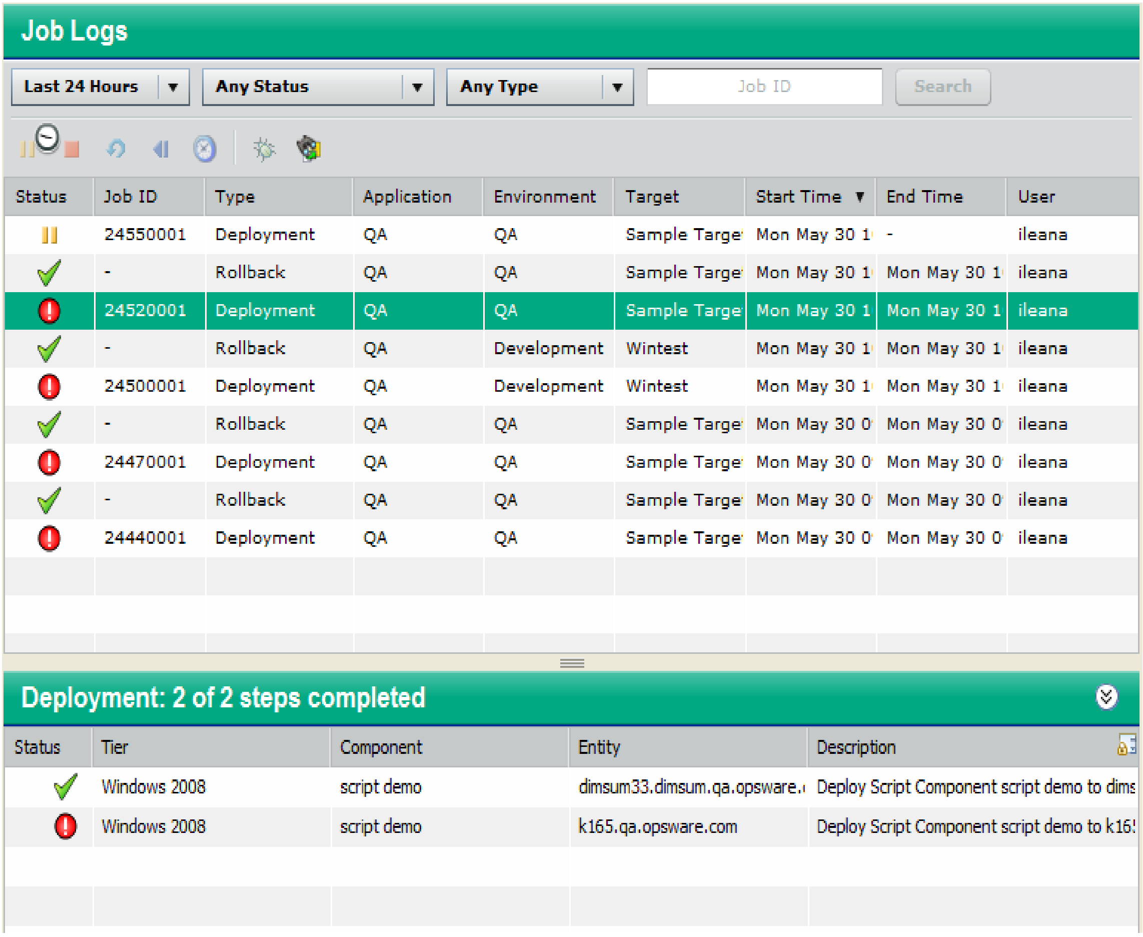
The following topics are discussed in this section:

- "Jobs screen" on the next page
- "Blocked jobs" on page 561

Jobs screen

The Jobs screen enables you to view a cumulative log of Application Deployment jobs. The following figure shows the right panel of the Jobs screen.

Jobs screen



Job Logs

Last 24 Hours ▼ | Any Status ▼ | Any Type ▼ | Job ID | Search

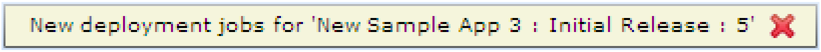
Status	Job ID	Type	Application	Environment	Target	Start Time ▼	End Time	User
‖	24550001	Deployment	QA	QA	Sample Targe	Mon May 30 1	-	ileana
✓	-	Rollback	QA	QA	Sample Targe	Mon May 30 1	Mon May 30 1	ileana
!	24520001	Deployment	QA	QA	Sample Targe	Mon May 30 1	Mon May 30 1	ileana
✓	-	Rollback	QA	Development	Wintest	Mon May 30 1	Mon May 30 1	ileana
!	24500001	Deployment	QA	Development	Wintest	Mon May 30 1	Mon May 30 1	ileana
✓	-	Rollback	QA	QA	Sample Targe	Mon May 30 0	Mon May 30 0	ileana
!	24470001	Deployment	QA	QA	Sample Targe	Mon May 30 0	Mon May 30 0	ileana
✓	-	Rollback	QA	QA	Sample Targe	Mon May 30 0	Mon May 30 0	ileana
!	24440001	Deployment	QA	QA	Sample Targe	Mon May 30 0	Mon May 30 0	ileana

Deployment: 2 of 2 steps completed

Status	Tier	Component	Entity	Description
✓	Windows 2008	script demo	dimsum33.dimsum.qa.opsware.	Deploy Script Component script demo to dims
!	Windows 2008	script demo	k165.qa.opsware.com	Deploy Script Component script demo to k16!

The Job Logs section contains a list of all the Application Deployment jobs that match the filter criteria that you specify. You can filter based on a certain time window, job status, job type, or a specific Job ID.

If you selected “Show job status when deployment starts” when you started a Deployment job, the Job Logs automatically gets filtered to show only your job. The following box appears at the top of the Job Logs area:



New deployment jobs for 'New Sample App 3 : Initial Release : 5' ✖

You can clear this filter by selecting the “Clear Search” button ✖ located at the top.

When you select a row in the Job Logs section, the steps for that job are shown in the lower section. In the example shown here, note the following.

- None of the steps have a check mark ✓ indicating that they were successfully completed.
- Steps 1 and 2 have failed: ❗
- Steps 3 and 8-13 are still scheduled: 🕒
- Steps 4 through 7 will be skipped: 🚫

If a single step in a Deployment job fails, the entire job is automatically rolled back, and the target servers are returned to the state they were in just prior to the deployment.

An automatic Rollback job is always created. If there is nothing to roll back, the job will have no steps, and its description will indicate why.

Job status indicators

Icon	Status	Meaning
✓	Succeeded	The step or job has completed successfully.
❗	Failed	The step or job has failed.
🕒	Scheduled	The step or job is scheduled to run.
🚫	Skip	This step will be skipped, because either the component or server to which it pertains was not selected in the debugger. See "Debugging a deployment job" on page 566.
▬	Skipped	This step was already skipped, because either the component or server to which it pertains was not selected in the debugger. See "Debugging a deployment job" on page 566.
▶	In Progress	The step or job is currently running. For a job with numerous steps, a green progress bar is displayed in the Status column as the steps are being executed.
⚠	Attention Required	The job has failed, and it is now in a suspended state to facilitate debugging. See "Debugging a deployment job" on page 566.

Job status indicators, continued

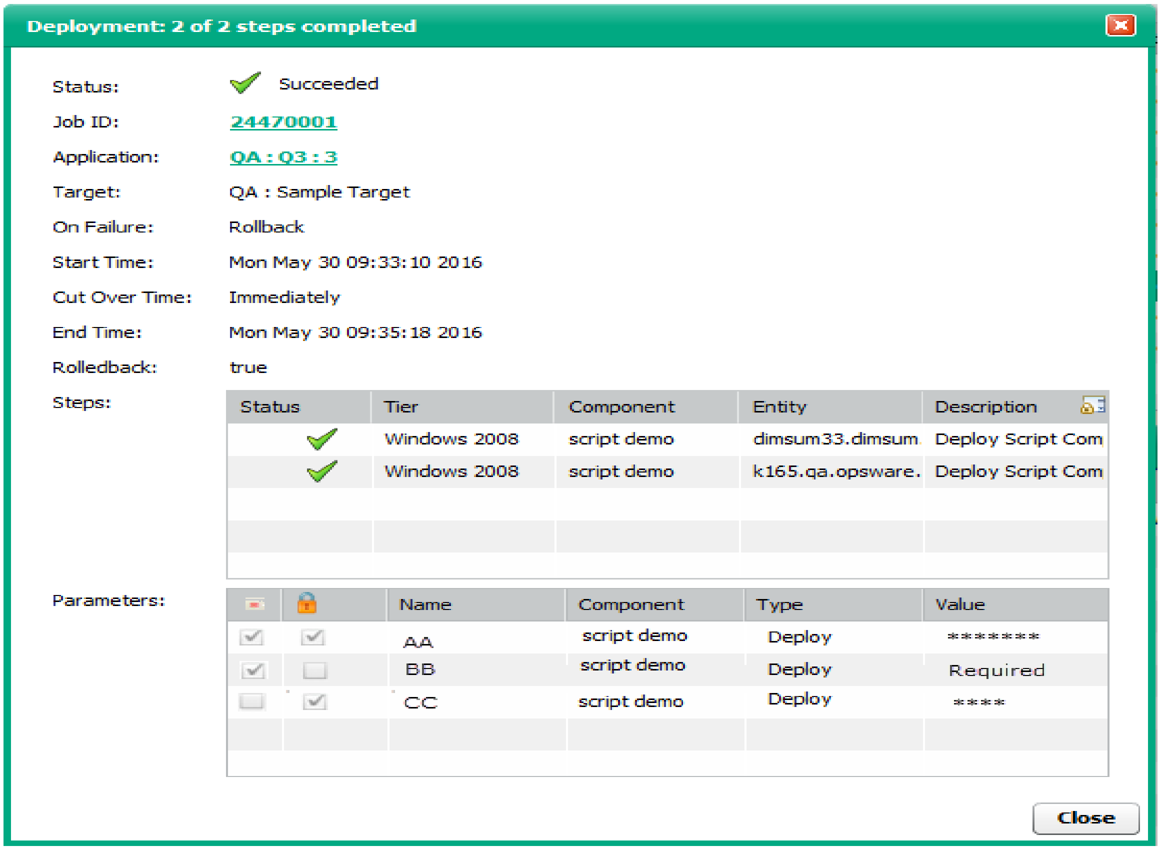
Icon	Status	Meaning
	Paused	The job has been paused.
Paused (pending)	The “Pause” button has been pressed, but the job has not yet paused. The steps that are currently executing must finish before the job can be paused.	The job has been cancelled.
🟥	Cancelled	
Cancelled (pending)	The “Cancel” button has been pressed, but the job has not yet been cancelled. The steps that are currently executing must finish before the job can be cancelled.	

If Application Deployment (the “da” component of SA) is stopped and then restarted while a job is In Progress, it carefully examines the status of each job step to determine the status of the job. If all job steps have successfully completed, the job status is set to Succeeded. If any job step is In Progress, that step will be marked Failed, an error will be logged to that step’s output, and the job will also be marked Failed.

By default, the first In Progress step of the job selected in the Job Logs table is highlighted. When that step finishes and a new step starts, the highlight moves to the new step. If you want to prevent the highlight from moving, you can use the “Scroll Lock” button.

To view more information about a job, double-click the row corresponding to that job in the Job Logs table. A new window opens, as shown here:

Job Detail window



Status:	✓ Succeeded
Job ID:	24470001
Application:	QA : Q3 : 3
Target:	QA : Sample Target
On Failure:	Rollback
Start Time:	Mon May 30 09:33:10 2016
Cut Over Time:	Immediately
End Time:	Mon May 30 09:35:18 2016
Rolledback:	true

Steps:

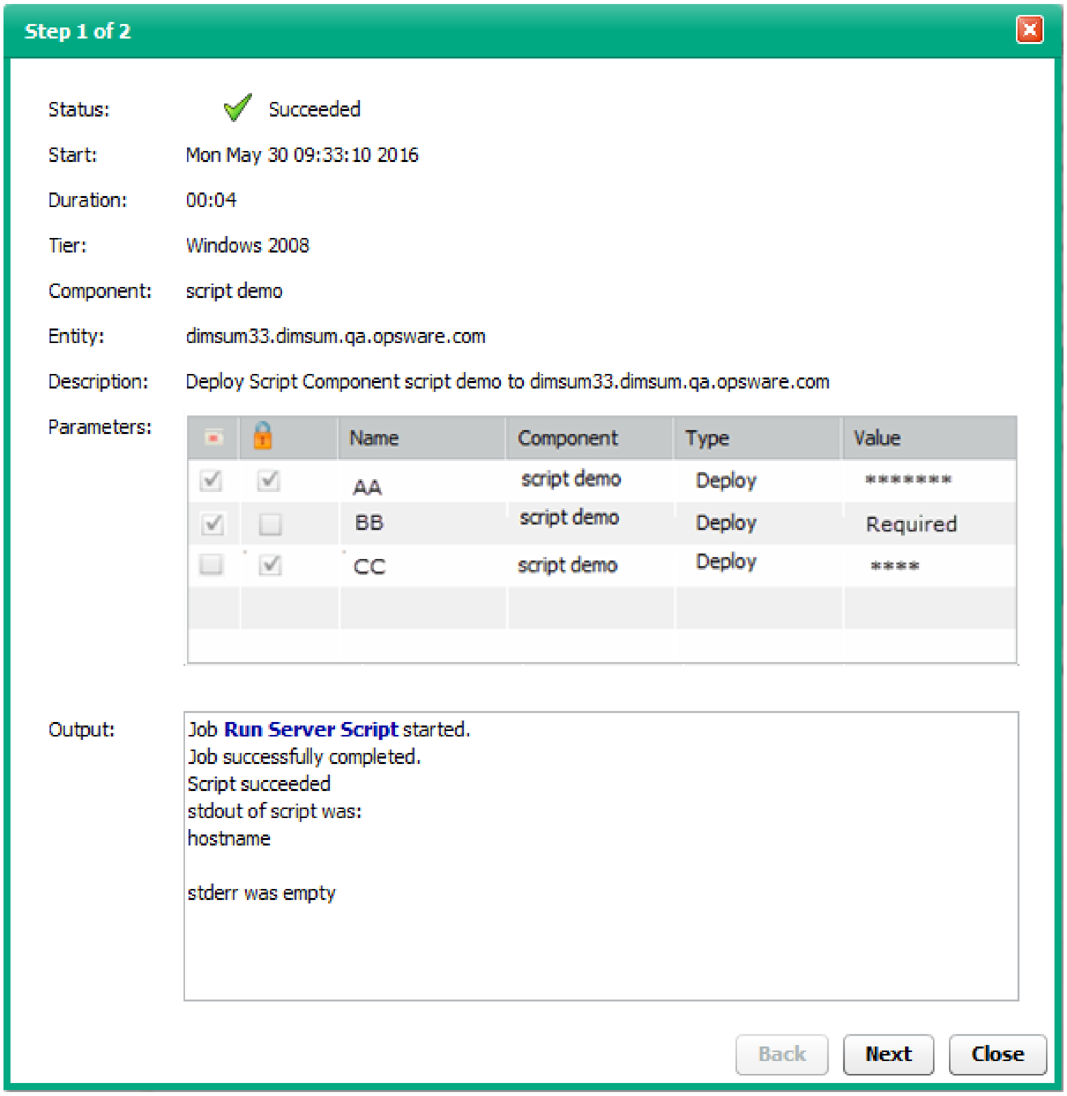
Status	Tier	Component	Entity	Description
✓	Windows 2008	script demo	dimsum33.dimsum	Deploy Script Com
✓	Windows 2008	script demo	k165.qa.opsware.	Deploy Script Com

Parameters:

		Name	Component	Type	Value
☑	☑	AA	script demo	Deploy	*******
☑	□	BB	script demo	Deploy	Required
□	☑	CC	script demo	Deploy	****

If you double-click a row in the Steps table, you can view more detail about how that step transpired. In the following example, the details for step 4 are shown.

Step Detail window



Status: ✓ Succeeded

Start: Mon May 30 09:33:10 2016

Duration: 00:04

Tier: Windows 2008

Component: script demo

Entity: dimsum33.dimsum.qa.opsware.com

Description: Deploy Script Component script demo to dimsum33.dimsum.qa.opsware.com

Parameters:

	🔒	Name	Component	Type	Value
☑	☑	AA	script demo	Deploy	*******
☑	□	BB	script demo	Deploy	Required
□	☑	CC	script demo	Deploy	****

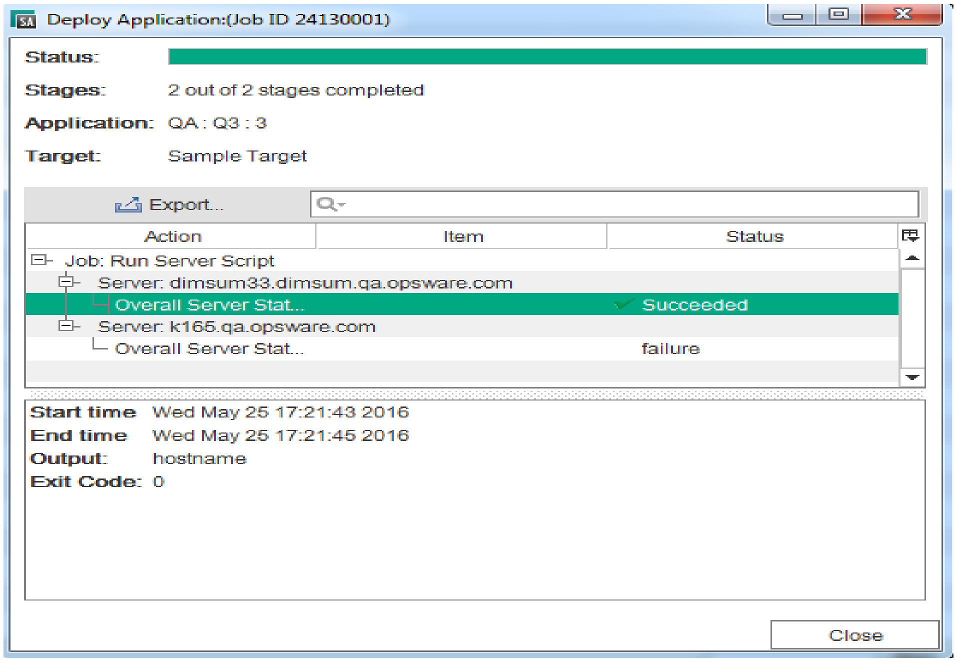
Output:

```
Job Run Server Script started.
Job successfully completed.
Script succeeded
stdout of script was:
hostname

stderr was empty
```

In this simple example, a Script component called “script demo” echoes the value of each of its three parameters to stdout. As shown in the Output box, the values of the four parameters were echoed to stdout, and the script successfully completed.

In the Step detail window, any text that appears in the Output box in a bold, blue font is a link to more detailed information presented in an SA “Deploy Application” job window. In the following example, the Run Server Script link was clicked.

Deploy Application Job window



Status:

Stages: 2 out of 2 stages completed

Application: QA : Q3 : 3

Target: Sample Target

Action	Item	Status
Job: Run Server Script		
Server: dimsum33.dimsum.qa.opsware.com		
Overall Server Stat...		✓ Succeeded
Server: k165.qa.opsware.com		
Overall Server Stat...		failure

Start time Wed May 25 17:21:43 2016
End time Wed May 25 17:21:45 2016
Output: hostname
Exit Code: 0

The information in this window is organized by server, and each server in the target is listed. Each “Overall Server Status” row corresponds to a single step that was performed on a particular server in the original Application Deployment job. Click one of these rows to view detailed information about the execution of that step on a specific server.

You can also open this window by clicking the Job ID in the Application Deployment Job detail window.

Blocked jobs

If Require Approval is selected for a particular type of job (on the Job Blocking page on the Administration tab in the SA Client), Application Deployment will display a message that the job is blocked pending approval.

A blocked job that is pending approval must be approved or cancelled on the Jobs and Sessions tab in the SA client. It cannot be cancelled from the Application Deployment Jobs screen.

If a blocked job is cancelled in the SA client, the Application Deployment Jobs screen will indicate that this job has failed.

For more information, see the Job Approval Integration section in "Developer guide overview" on page 23.

Working with jobs

You can use the Jobs window to do any of the following things:

- "Locating specific jobs" below
- "Pausing a job" on the next page
- "Resuming a paused job" on page 564
- "Canceling a job" on page 564
- "Rescheduling a job" on page 565
- "Rolling back a deployment" on page 565
- "Undeploying a deployment" on page 566

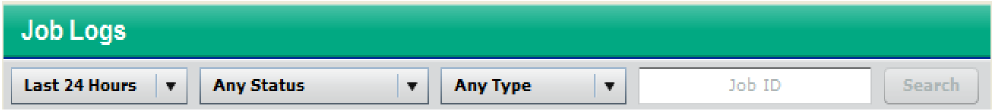
You can also use the Application Deployment Debugger to troubleshoot a job. See "Debugging a deployment job" on page 566 for more information.

For information about drilling down to detailed information from the Jobs screen, see "Jobs screen " on page 555.

Locating specific jobs

You can use the Search tool to filter the jobs displayed in the Job Logs panel or to locate a specific job (if you know the Job ID).

Jobs Screen Search tool



In this example, only those Deployment jobs that had failed during the Last Week would be listed.

To filter the Job logs:

1. Go to the Jobs screen (click Jobs in the lower left corner).
2. Do one or more of the following:
   - Specify the time window
   - Specify the job status
   - Specify the job type
   - Specify a Job ID
3. Click **Search**.

When searching by Job ID, you must specify the entire Job ID (no partial IDs or wildcard characters). You can specify multiple Job IDs separated with commas.

To broaden the search and display more jobs, select the “Any” choice in one or more of the drop-down lists.

Pausing a job

You can pause any job that you have deployed that is currently in progress. The Application Deployment administrator can pause any job that is currently in progress.

When you pause a job, Application Deployment will complete the step that it is currently processing and then pause. Until the current step finishes, the status of the job will be “Paused (pending).” After the current step finishes, the status of the job will be “Paused” until it is either cancelled or instructed to resume.

To pause a job:

1. Go to the Jobs screen (click Jobs in the lower left corner).
2. Locate and select the job that you want to pause in the Job Logs (see "Locating specific jobs" on the previous page).
3. On the tool bar, click **Pause**.
4. Click **Yes** to confirm.

The job will remain paused until you (or the Application Deployment administrator) resume the job (see "Resuming a paused job" on the next page).

Resuming a paused job

You can resume any job that you have paused. The Application Deployment administrator can resume any paused job.

When you resume a job, Application Deployment starts running the next scheduled step in the job.

To resume a paused job:

1. Go to the Jobs screen (click Jobs in the lower left corner).
2. Locate and select the paused job that you want to resume in the Job Logs (see "Locating specific jobs" on page 562).
3. On the tool bar, click **Resume**.

The job will continue running from the point at which it was previously paused (see "Pausing a job" on the previous page).

Canceling a job

You can cancel any job that you have deployed that is currently either in progress or paused. The Application Deployment administrator can cancel any job that is in progress or paused.

When you cancel a job, Application Deployment will complete the step that it is currently processing and then stop. Until the current step finishes, the status of the job will be “Cancelled (pending).” After the current step finishes, the status of the job will be “Cancelled.” Any remaining steps are left in the “Scheduled” state.

If the job that you are cancelling is a Deployment job, Application Deployment will attempt to roll back the deployment.

To cancel a job:

1. Go to the Jobs screen (click Jobs in the lower left corner).
2. Locate and select the job that you want to cancel in the Job Logs (see "Locating specific jobs" on page 562).
3. On the tool bar, click **Cancel**.
4. Click **Yes** to confirm.

A cancelled job cannot be resumed. If you later decide to deploy this version to the same target, you can simply deploy the same version again.

Rescheduling a job

You can reschedule a “Scheduled” job that you have deployed. The Application Deployment administrator can reschedule any “Scheduled” job.

To reschedule a job:

1. Go to the Jobs screen (click Jobs in the lower left corner).
2. Locate and select the job that you want to reschedule in the Job Logs (see "Locating specific jobs" on page 562). It must be in the “Scheduled” state.
3. On the tool bar, click **Reschedule**.
4. Choose a new Stage time, a new Cut Over time, or both.
5. Click **OK**.

Note the following:

If a job has been scheduled but is not yet running, and the job has “auto-undeploy” enabled, you will get an information when you reschedule the job indicating that you will need to reschedule both the Undeployment and Deployment jobs.

If Staging has happened, but Cut Over has not yet happened, you can reschedule only the Cut Over segment.

The **Reschedule** button is disabled when the job is running.

Rolling back a deployment

After a Deployment job that you have deployed has completed, you can manually roll it back. This returns the server to the state it was in prior to the deployment. The Application Deployment administrator can roll back any Deployment job.

See "Rolling back a deployment" on page 552 for instructions.

If a job fails or is cancelled, and rollback is enabled, Application Deployment will roll it back automatically.

The standard Rollback scripts maintain only one backup version for each release—they remove any existing backup files in order to conserve disk space on the target server. This means that if you roll back a version that is older than the most recently deployed version, Application Deployment will not be able to restore the code files. If you attempt to roll back a previous version, a warning message is displayed, and you can either confirm or cancel the rollback.

Undeploying a deployment

After a Deployment job that you have deployed has completed, you can manually undeploy it. This removes any trace of the deployment, but it does not attempt to return the server to the state it was in prior to the deployment. If no intervening versions have been deployed, the undeploy is equivalent to a rollback. The Application Deployment administrator can undeploy any Deployment job.

An undeploy is a more aggressive type of “undo” operation than a rollback.

Debugging a deployment job

You can use the Application Deployment debugger to troubleshoot a Deployment job, a Rollback job, or an Undeploy job (you cannot use it to debug a Make Version job).

Consider the following scenario:

1. A Deployment job fails because of a problem in a specific component.
2. You make a change to fix that component.
3. You then create a debuggable job and launch the debugger.
4. In the debugger, you disable as much of that job as possible—this enables you to quickly test your change and see if it works.
5. If your change works, you can then enable more of the job until you are confident that the entire job works.

Depending on your troubleshooting objective, you can launch the debugger from either the Deployment screen or the Jobs screen. You can use the debugger to do the following things:

- Use breakpoints
- Deploy or skip any component
- Deploy to or skip any server
- Single-step through a deployment
- Enable or disable tier provisioning and backup steps

When you skip a component or server, all job steps pertaining to that component or server—including rollback and undeployment steps—are skipped.

To run the debugger, you must either have Edit permission for the pertinent application (see "Setting permissions" on page 579) or Manage Application Deployment permission (see "Setting permissions" on page 579).

Debugger window

The debugger runs in separate window that has seven functional areas:

- "Tool bar" below
- "Status bar" on the next page
- "Job options" on the next page
- "Job selections" on page 569
- "Details" on page 570
- "Steps" on page 570
- "Parameters" on page 571
- "Output" on page 571

To expand an area that is collapsed, click the “Expand” ⌄ button on the blue bar.

For information about performing common troubleshooting tasks using the debugger, see "Troubleshooting a job" on page 572.

Tool bar

The debugger tool bar, located in the upper left corner of the Job Debugger window, enables you to control how the job that you are troubleshooting runs.

Debugger Tool Bar Buttons

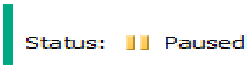
Button	Name	Purpose
↳☰	Single Step	Click this button to run the next step in the debug job. Steps that are marked “Skipped” 🚫 are not executed.
▶	Resume	Click this button to continue running a Paused job.

Debugger Tool Bar Buttons, continued

Button	Name	Purpose
		If a breakpoint has been set, the job will run until it reaches the breakpoint.
⏸	Pause	Click this button to pause the debug job. The Pause button replaces the Resume button when a job is running.
🟥	Cancel Job	Click this button to cancel the entire debug job.
◉	Set Breakpoint	Select a “Scheduled” ⊗ step in the "Steps" on page 570 area, and then click this button to set a breakpoint at this step. The job will stop before executing the step with the breakpoint. You cannot set breakpoints on steps that have already been executed or steps that will be skipped.
☒	Remove Breakpoint	Select a step with a breakpoint in the "Steps" on page 570 area, and then click this button to remove the breakpoint.
🔄	Refresh	Force a refresh of the entire Job Debugger window.

Status bar

The status bar is located in the lower left corner of the Job Debugger window. It shows you the current status of the debug job:

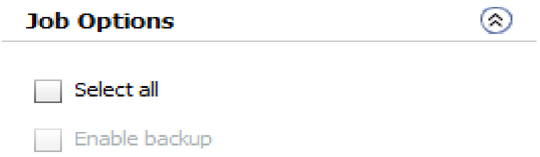


Status: ⏸ Paused

See "Job status indicators" on page 557 for a description of all possible job states.

Job options

The Job Options area is located just below the tool bar in the upper left corner of the Job Debugger window. Here, you can set the following options:



Job Options

□ Select all

□ Enable backup

The “Select all” option selects all servers and all components included in this job. If you clear the “Select all” box, all jobs steps will be skipped. See "Selecting or skipping specific components or target items" on page 573 for additional information.

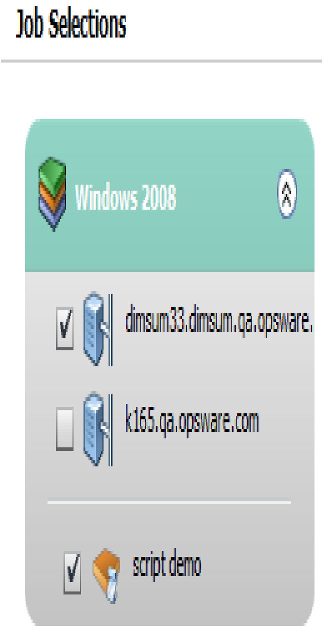
The “Enable backup” option runs any automatic backup steps that are included in this job. To disable all backup steps, clear this box. This will make the debug job run faster, but it will not roll back properly.

The “Enable backup” box is automatically cleared if you clear the “Select all” box—because clearing the “Select all” box skips all job steps, including backup steps. See "Skipping backup steps" on page 577for more information.

These options are only available before any steps in the debug job have been executed.

Job selections

The Job Selections area is located on the left side of the Debugger window:



Here, you can select individual servers and components to include in your debug job. When you clear the box corresponding to a specific component or server, any steps pertaining to that component or server—including any rollback and undeployment steps—will be skipped.

In the example above, all steps pertaining to server `k165.opsware.com` will be skipped.

You can also enable or disable the deployment of a Software Policy or OS Sequence associated with any tier (see the SA 10.50 Administration Guide).

Components or servers for which steps have already been either executed or skipped are disabled in the Job Selections area and cannot be selected.

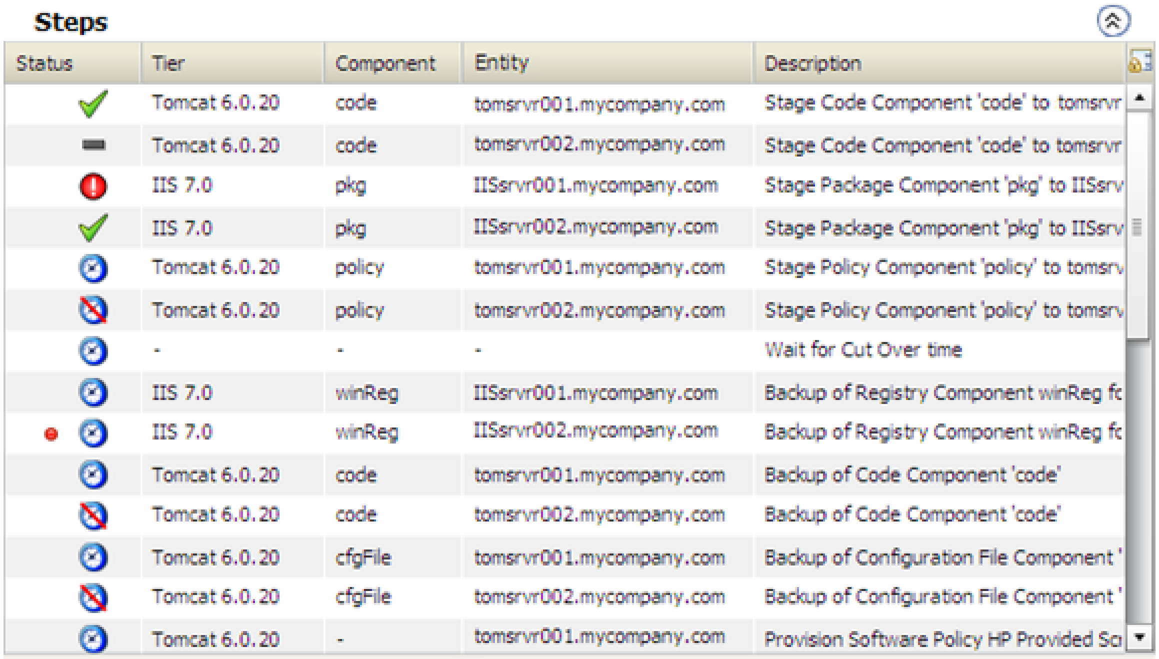
Details

The Details area is located in the upper right corner of the Job Debugger window. It contains information about the job that you are troubleshooting. The Job ID, Target, and Start Time fields are not populated until the job runs. If you launch the debugger from the Deployment page, for example, these fields will be empty until the job runs at least one executable step.

You can add or modify the comment in the Details area. This comment appears in the details window when you double-click this job in the Job Logs area of the Jobs screen.

Steps

The Steps area is located on the right side of the Job Debugger window below the Details area:



Steps

Status	Tier	Component	Entity	Description
✓	Tomcat 6.0.20	code	tomsrvr001.mycompany.com	Stage Code Component 'code' to tomsrvr
▬	Tomcat 6.0.20	code	tomsrvr002.mycompany.com	Stage Code Component 'code' to tomsrvr
❗	IIS 7.0	pkg	IISsrvr001.mycompany.com	Stage Package Component 'pkg' to IISsrv
✓	IIS 7.0	pkg	IISsrvr002.mycompany.com	Stage Package Component 'pkg' to IISsrv
◷	Tomcat 6.0.20	policy	tomsrvr001.mycompany.com	Stage Policy Component 'policy' to tomsrv
⊘	Tomcat 6.0.20	policy	tomsrvr002.mycompany.com	Stage Policy Component 'policy' to tomsrv
◷	-	-	-	Wait for Cut Over time
◷	IIS 7.0	winReg	IISsrvr001.mycompany.com	Backup of Registry Component winReg fc
● ◷	IIS 7.0	winReg	IISsrvr002.mycompany.com	Backup of Registry Component winReg fc
◷	Tomcat 6.0.20	code	tomsrvr001.mycompany.com	Backup of Code Component 'code'
⊘	Tomcat 6.0.20	code	tomsrvr002.mycompany.com	Backup of Code Component 'code'
◷	Tomcat 6.0.20	cfgFile	tomsrvr001.mycompany.com	Backup of Configuration File Component '
⊘	Tomcat 6.0.20	cfgFile	tomsrvr002.mycompany.com	Backup of Configuration File Component '
◷	Tomcat 6.0.20	-	tomsrvr001.mycompany.com	Provision Software Policy HP Provided Sc

The Steps area shows you each individual step that Application Deployment performs when it runs this job. When you clear the check box for a server or component in the "Job selections" on the previous page area, all steps that pertain to that server or component are skipped.

In the example above, all steps pertaining to server `tomsrvr002.mycompany.com` will be skipped.

The status icons are updated as the steps are executed.

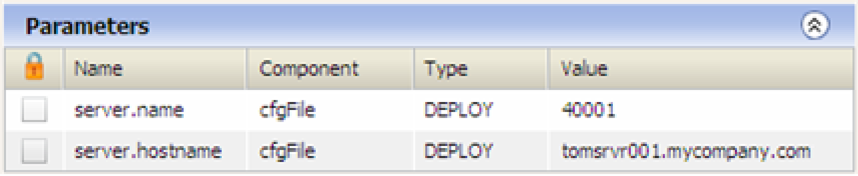
Debugger Step Status Indicators

Icon	Status	Description
🕒	Scheduled	This step is scheduled to run.
⊘	Skip	This step will be skipped (because either the component or server to which it pertains is not selected in the "Job selections" on page 569 area).
▶	In Progress	This step is running.
✓	Succeeded	This step succeeded.
❗	Failed	This step failed.
▬	Skipped	This step was already skipped.

Parameters

The first time that you open the debugger for a specific job, and no step has yet been selected, the Parameters table lists all the parameters for that job. If the job has no parameters, the message “Job has no parameters” is displayed.

When you select a specific step, the Parameters table lists only those parameters that pertain to the component being deployed in that step:



Parameters

🔒	Name	Component	Type	Value
□	server.name	cfgFile	DEPLOY	40001
□	server.hostname	cfgFile	DEPLOY	tomsrvr001.mycompany.com

If the step has no parameters, the message “Step has no parameters” is displayed.

The Parameters table is read-only. You cannot override parameter values in the debugger.

Output

The Output area shows you what Application Deployment did when it ran the step selected in the "Steps" on the previous page area. It indicates whether the step finished successfully and displays the contents of `stdout` and `stderr` (if pertinent).

Any information that appears in bold blue text in the Output area is a link to more detailed information

Troubleshooting a job

The topics in this section show you how to perform the following troubleshooting tasks using the debugger:

- "Launching the debugger" below
- "Selecting or skipping specific components or target items" on the next page
- "Pausing a debug job" on page 575
- "Resuming a debug job" on page 575
- "Single-stepping a debug job" on page 575
- "Working with breakpoints" on page 575
- "Viewing detailed information about a step " on page 576
- "Adding a comment to a deployment " on page 576
- "Skipping backup steps" on page 577
- "Skip tier policy or provisioning" on page 577
- "Canceling a debug job" on page 578
- "Debugging a rollback or undeployment job" on page 578

See for information about navigating the Job Debugger window, see "Debugger window" on page 567.

Launching the debugger

There are two ways to launch the debugger:

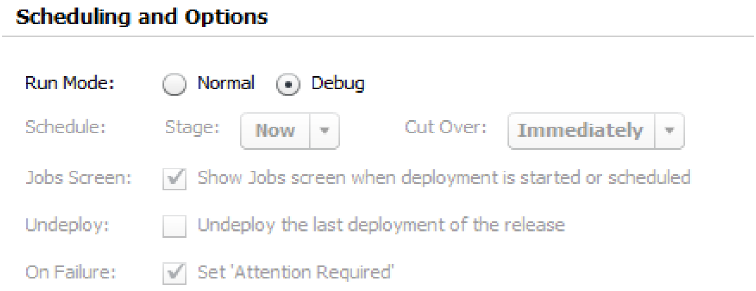
- If you want to troubleshoot a Deployment job from the beginning, launch the debugger from the Deployment screen.
- If you want to troubleshoot a job that is already In Progress or Paused, launch it from the Jobs screen.

  From the Jobs screen, you can launch the debugger for Deployment jobs, Rollback jobs, and Undeploy jobs (you cannot debug Make Version jobs).

You can only launch the debugger for jobs that are in the Paused, In Progress, or Attention Required state. You cannot launch the debugger for a Scheduled, Completed, Cancelled, or Failed job.

To launch the debugger from the Deployment screen:

1. Under Scheduling and Options, select **Debug** mode.



Scheduling and Options

Run Mode: ○ Normal ● Debug

Schedule: Stage: Now Cut Over: Immediately

Jobs Screen: ☑ Show Jobs screen when deployment is started or scheduled

Undeploy: □ Undeploy the last deployment of the release

On Failure: ☑ Set 'Attention Required'

   The deployment schedule and options are automatically set when you select **Debug** mode.

   For more information about these options, see "Scheduling and options" on page 540.

2. Click **Start**.

To launch the debugger from the Jobs screen:

1. In the right pane, select the job that you want to debug.
2. Click **Debug** on the tool bar.

Selecting or skipping specific components or target items

When a debug job is in the Paused state, you can use the Job Selections settings to specify which components will be deployed to which target servers when the job resumes. You can deploy any, all, or none of the components in a tier to any, all, or none of the target items in that tier.

You can only change the setting for a particular component before any steps involving that component have started. Similarly, you can only change the setting for a particular server before any steps involving that target item have started

When you launch the debugger by selecting “Debug” mode on the Deployment page (see "Launching the debugger" on the previous page), your job is automatically placed in the Paused state before any steps are executed. In this case, you have full control over which components and target items are included or excluded.

When you launch the debugger from the Jobs screen, you will only have access to the Job Selections settings if one of the following conditions is true:

- The job was in the Paused state before you clicked **Debug** on the Jobs screen.
- You clicked **Pause** in the debugger.

In either case, you will only be able to change the settings for a component or target item if no steps involving that component or target item have started.

The following instructions show you how to implement some common debugging scenarios. All of these examples assume that you have launched the debugger by selecting “Debug” mode on the Deployment page.

To deploy all components to all target items in all tiers:

In the Job Options area, select the **Select all** box.

To skip all steps pertaining to a particular component or target item:

In the Job Selections area, clear the box for the component or target server that you want to skip.

All steps involving that component or target item will show “Skip” ⊘ status in the Steps area.

To deploy a single component to all target items in a tier:

1. In the Job Options area, clear the **Select all** box.
2. In the Job Selections area, select the component that you want to deploy. All servers in the tier are automatically selected.

To deploy all pertinent components to a single server:

1. In the Job Options area, clear the **Select all**box.
2. In the Job Selections area, select the server where you want to deploy the components. All components in the tier are automatically selected.

To deploy a single component to a single server:

1. In the Job Options area, clear the **Select all** box.
2. In the Job Selections area:
   a. Select the component that you want to deploy. All servers in the tier are automatically selected.
   b. Clear the boxes for all of the servers tier except the server where you want to deploy this component.

Pausing a debug job

When a debug job is in the In Progress state, and you want to change the "Job selections" on page 569 settings to include or skip individual components or servers, you must first pause the job.

When a debug job is in the Paused state, the "Troubleshooting a job" on page 572 settings are enabled, and you can change the setting for any component or server for which no steps have been either executed or skipped.

To pause a debug job that is in the In Progress state, click the “Pause” ⏸ button on the tool bar.

The job will first transition to the Pause-Pending state (to finish executing any steps currently in progress), and then it will transition to the Paused state.

Resuming a debug job

To resume a debug job that is in the Paused state, click **Resume** on the tool bar.

If you click the “Resume” button for a debug job that is in the Attention Required state, a message appears warning you that the job will transition to the Failed state if you resume. You must explicitly click OK to resume the job. Application Deployment will then set the status of the original Deployment job to Failed and initiate a Rollback job (if applicable).

Single-stepping a debug job

To single-step a debug job that is in the Paused state, click **Single Step** on the tool bar. The next scheduled (not skipped) step is executed.

If you click **Single Step** button for a debug job that is in the Attention Required state, a message appears warning you that the job will transition to the Failed state if you proceed to single-step. You must explicitly click OK to single-step the job. Application Deployment will then set the status of the original Deployment job to Failed and initiate a Rollback job (if applicable).

Working with breakpoints

When a debug job is in the Paused state, you can set a breakpoint at any scheduled (not skipped, not yet executed) step. When you resume the job, the debugger will run all scheduled (not skipped) steps prior to that breakpoint and then pause.

Breakpoints are persistent within an Application Deployment user interface (UI) session. If you exit the Application Deployment UI and then open it again later, you will lose any breakpoints that you set in the previous session.

After a job has completed and is either in the Succeeded or Failed state—where it can no longer be debugged—any breakpoints are automatically removed.

To set a breakpoint:

1. In the Steps area, select the step where you want to set the breakpoint.
2. Click **Set Breakpoint** on the tool bar.

You cannot set a breakpoint on a step that has already started.

To remove a breakpoint:

1. In the Steps area, select the step with the breakpoint.
2. Click the **Remove Breakpoint** on the tool bar.

You cannot remove a breakpoint from a step that has already started.

Viewing detailed information about a step

When you select (single-click) a step in the "Steps" on page 570 area, the "Parameters" on page 571 and "Output" on page 571 areas show information pertaining to that step.

When you double-click a step in the "Steps" on page 570 area, detailed information about that step is displayed in a pop-up window. See the "Step Detail window" on page 559 for a summary of the information displayed.

Adding a comment to a deployment

You can add or modify the comment associated with a debug job. This is useful if you want to be able to distinguish debug jobs in the Jobs screen by using the contents of the comment field. You can add a comment regardless of the state of the debug job.

To add a comment:

1. In the debugger, expand the Details area.
2. Type a comment in the **Comment** box.

3. Click **Save**.
4. If you want to undo any changes to the comment, click **Reset**.

Skipping backup steps

If you launch the debugger from the Deployment screen by selecting “Debug” mode (see "Launching the debugger" on page 572), you can enable or disable any backup steps in the debug job. By default, the backup steps are enabled. Disabling these steps can save time during debugging.

Not all component types have backup steps. For example: Script, Application Configuration, and OO Flow components do not have backup steps. If a debug job has no components with backup steps, the “Enable backup” is disabled.

You can only change the Job Options settings before any steps have been executed. After any step in the job begins running, you cannot change these settings.
If you launch the debugger by clicking **Debug** on the Jobs page, the settings in the "Job options" on page 568 area are disabled. This is because some steps have already been executed by the time that you enter the debugger.

To disable all backup steps:

In the Job Options area, clear the “Enable backup” box.

Skip tier policy or provisioning

If you launch the debugger from the Deployment screen by selecting “Debug” mode (see "Launching the debugger" on page 572), you can enable or disable the Software Policy or OS Sequence associated with any tier. These steps are typically time-consuming and may not be necessary, depending on where the deployment that you are debugging is failing.

You can only change the tier settings when the debug job is in the Paused state and before any tier provisioning steps pertaining to that tier have been executed.

To disable tier policy or OS provisioning:

1. In the Job Selections area, clear the Tier Policy/OS Provision box for each pertinent tier.

   Only those tiers that have a Software Policy, an OS Sequence, or both will have this setting (see "Administering tiers" on page 593).

2. Using the tool bar buttons, resume or single-step the deployment job.

Canceling a debug job

You can cancel a debug job that is either running or paused.

To cancel a debug job:

1. In the debugger, click **Cancel** on the tool bar.
2. In the Cancel Job dialog box, click **Yes**.

   At this point, the following things happen:

   - The job transitions to the Cancel-Pending state (to finish executing any steps currently in progress).
   - The debug job is terminated.
   - The original Deployment job is placed in the Cancelled state.
   - The original Deployment job is rolled back (if applicable).

Debugging a rollback or undeployment job

You can use the debugger to troubleshoot a Rollback or Undeployment job. The process is similar to debugging a Deployment job.

To debug a rollback or undeployment job:

1. On the Jobs screen, select a Rollback or Undeploy job that is In Progress.
2. Click **Pause** to pause the job.
3. After the job transitions to the Paused state, click **Debug** to launch the debugger.

The same constraints apply to debugging Rollback and Undeployment jobs. Because you are launching the debugger after job steps have started, the "Job options" on page 568 settings are not available. You cannot change the "Job selections" on page 569 settings for components or servers for which steps have already been executed or skipped.

Setting permissions

Overview

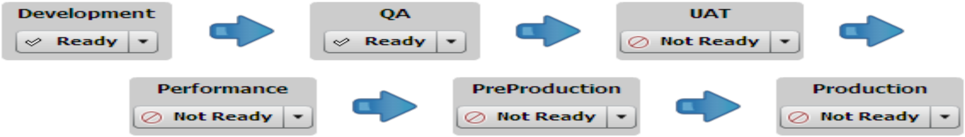
Application Deployment provides very finely grained control over two things:

- Who can create, view, edit, or deploy applications
- Who can use various library items in application components
- Where they can deploy them

Permissions are the mechanism used to establish this control. Permissions can help you promote an application through the software development life cycle in an orderly manner. The most convenient way to administer permissions is to create user groups that have a specific set of permissions.

Because Application Deployment offers very finely grained permissions, users can be granted permission to deploy a specific application to a specific environment—and nothing more. Their capabilities may appear escalated in comparison to their SA capabilities in general, but they are actually quite constrained.

Consider the following simple Patch life cycle:



For the purpose of developing and releasing a simple application or patch, for example, you might create the following user groups:

- "Application owners" on the next page
- "QA environment owners" on the next page
- "QA engineers" on page 581
- "Production environment owners" on page 582
- "Production engineers" on page 582

Each group would have specific capabilities based on the combination of permissions granted.

Application owners

Members of an Application Owners group would perform the following types of tasks:

- Create applications
- Attach Application Configurations to applications
- Attach Software Policies to applications
- Attach Packages to applications
- Deploy applications to the Development environment

To perform these tasks, Application Owners would need the following permissions:

Type of permission	Permission required
Action Permissions	System Administration: Managed Servers and Groups Application Deployment: Access Application Deployment Application Deployment: Create Applications Application Configuration: Manage Application Configurations (Read) Package Management: Manage Package (Read) Policy Management: Manage Software Policy (Read)
Resource Permissions	Read access to the device group that contains the Development environment servers
Folder Permissions	List and Read permission for Application Configurations, Packages, and Software Policies
Application Permissions	The application owner can grant View, Edit, and Deploy permissions for that application to other users and groups
Environment Permissions	Deploy permission for the Development environment

QA environment owners

Members of a QA Environment Owners group would perform the following types of tasks:

- Create targets
- Add servers to targets in the QA environment

To perform these tasks, QA Environment Owners would need the following permissions:

Type of permission	Permission required
Action Permissions	System Administration: Managed Servers and Groups Application Deployment: Access Application Deployment
Resource Permissions	Read access to the device group that contains the QA environment servers
Folder Permissions	None
Application Permissions	None
Environment Permissions	Edit permission for the QA environment

QA engineers

Members of a QA Engineers group would perform the following types of tasks:

- Deploy applications to the QA environment
- Troubleshoot deployment jobs

To perform these tasks, QA Engineers would need the following permissions:

Type of permission	Permission required
Action Permissions	System Administration: Managed Servers and Groups Application Deployment: Access Application Deployment
Resource Permissions	Read access to the device group that contains the QA environment servers
Folder Permissions	None
Application Permissions	Deploy permissions for the pertinent applications
Environment Permissions	Deploy permission for the QA environment

Production environment owners

Members of a Production Environment Owners group would perform the following types of tasks:

- Create targets
- Add servers to targets in the Production environment

To perform these tasks, Production Environment Owners would need the following permissions:

Type of permission	Permission required
Action Permissions	System Administration: Managed Servers and Groups Application Deployment: Access Application Deployment
Resource Permissions	Read access to the device group that contains the Production environment servers
Folder Permissions	None
Application Permissions	None
Environment Permissions	Edit permission for the Production environment

Production engineers

Members of a Production Engineers group would perform the following types of tasks:

- Deploy applications to the Production environment
- Troubleshoot deployment jobs

To perform these tasks, Production Engineers would need the following permissions:

Type of permission	Permission required
Action Permissions	System Administration: Managed Servers and Groups Application Deployment: Access Application Deployment
Resource Permissions	Read access to the device group that contains the Production

, continued

Type of permission	Permission required
	environment servers
Folder Permissions	None
Application Permissions	Deploy permissions for the pertinent applications
Environment Permissions	Deploy permission for the Production environment

Types of permissions

There are various types of permissions that affect what you can see and access in the Application Deployment user interface (UI):

"Action permissions" on the next page

Three types of Action permissions affect access to Application Deployment:

- The Managed Servers and Groups permission determines whether you can view managed servers and device groups. You need this capability in order to add servers to Application Deployment targets.
- The Application Deployment permissions determine whether you can access the Application Deployment UI, whether you can create applications, and whether you have Application Deployment administrator privileges.
- Three additional permissions determine whether you can reference Software Policies, Packages, Application Configurations, or OS Sequences in application components or attach them to tiers.

All Action permissions are set by the SA system administrator.

"Device group permissions" on page 587

These permissions determine which device groups you can access. You need access to a device group in order to add servers in that device group to Application Deployment targets.

Device Group permissions are set by the SA system administrator.

"Library folder permissions" on page 588

Folder permissions further determine which specific library items (Software Policies, Packages, Application Configurations, or OS Sequences) you can reference in your applications or attach to your tiers.

Folder permissions are set by the SA system administrator.

"Application Deployment permissions" on page 589

There are two types of Application Deployment permissions:

- Application permissions determine who is allowed to view, edit, or deploy a specific application.

  Application permissions are set by the user who creates the application—or by any user who has been granted Edit permission for that application. The Application Deployment administrator can also set application permissions.
- Environment permissions determine who is allowed to deploy applications into a specific environment. They also determine who is allowed to create or modify Application Deployment targets in that environment. Environment permissions are set by your Application Deployment administrator.

A description of the minimum permissions needed to accomplish common Application Deployment tasks is provided in this section. For a more comprehensive discussion of permissions, see the SA 10.50 Administration Guide.

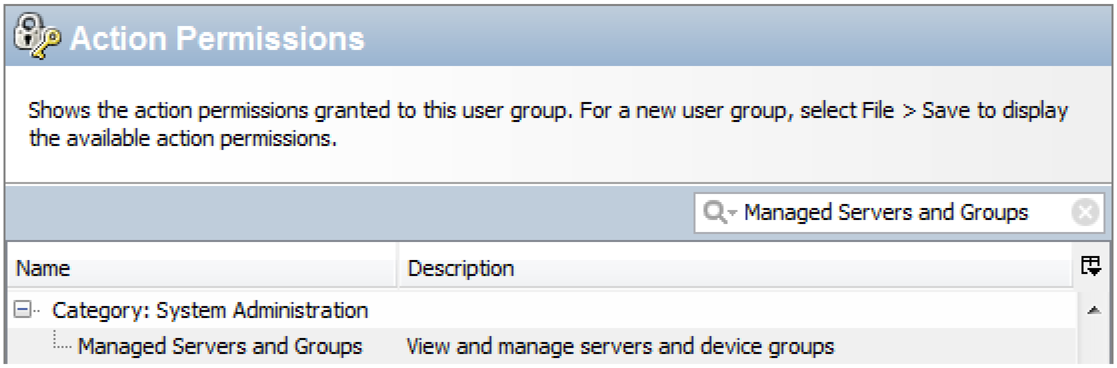
Action permissions

The following SA Action permissions are pertinent to Application Deployment.

Action permissions are granted to user groups by the SA system administrator using the Administration screen in the SA client (for more information, see the SA 10.50 Administration Guide.

Managed servers and groups permissions

If you need to create or modify Application Deployment targets, you must have the ability to view and manage servers and device groups:



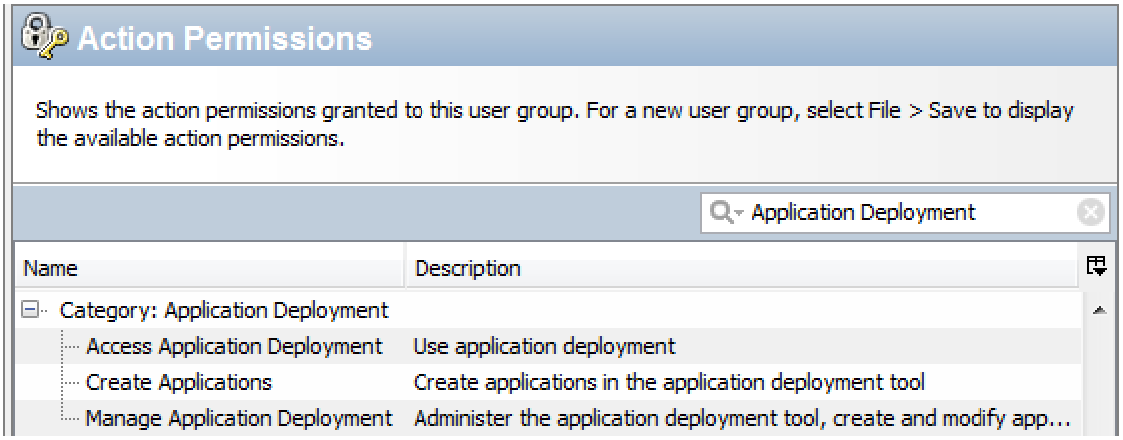
Shows the action permissions granted to this user group. For a new user group, select File > Save to display the available action permissions.

Managed Servers and Groups

Name	Description
Category: System Administration	
Managed Servers and Groups	View and manage servers and device groups

Application deployment access permissions

The following permissions determine whether you can access the Application Deployment UI and what you can do within it:



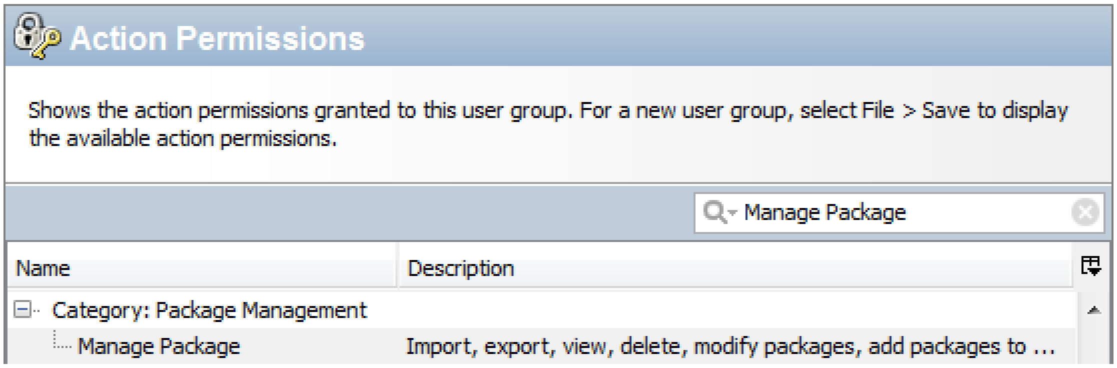
Shows the action permissions granted to this user group. For a new user group, select File > Save to display the available action permissions.

Name	Description
Category: Application Deployment	
Access Application Deployment	Use application deployment
Create Applications	Create applications in the application deployment tool
Manage Application Deployment	Administer the application deployment tool, create and modify app...

- Access Application Deployment enables you to launch Application Deployment. If you do not have this permission, the Application Deployment menu item will not appear on the Tools menu, and you cannot log in to Application Deployment using a web browser.
- Create Applications enables you to create applications in the Application Deployment user interface. Each application also has its own View/Edit/Deploy permissions (see "Application permissions" on page 589).
- Additional permissions are required to deploy applications into specific environments (see "Environment permissions" on page 591 and "Device group permissions" on page 587).
- Manage Application Deployment enables you to access the Administration screen in the Application Deployment UI. It also enables you to create and modify any application, target, or environment.

Package permissions

You must have Manage Package (Read) permission to add reference a software package in an application component or attach it to a tier.

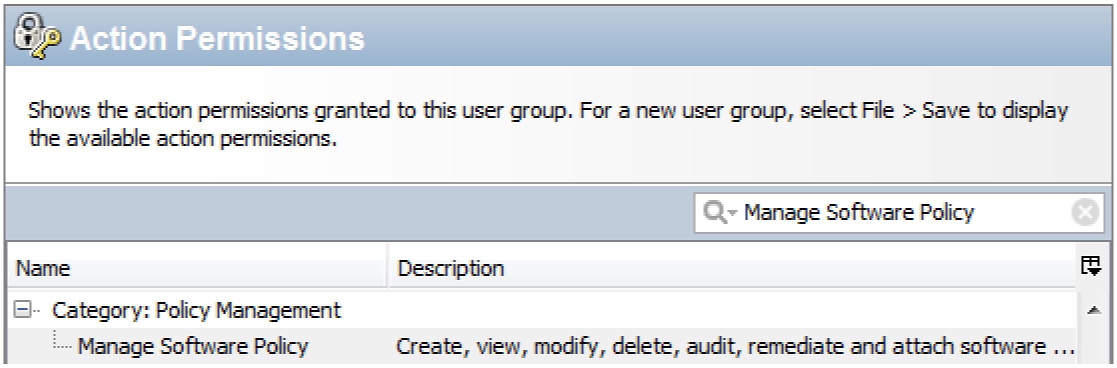


Shows the action permissions granted to this user group. For a new user group, select File > Save to display the available action permissions.

Name	Description
Category: Package Management	
Manage Package	Import, export, view, delete, modify packages, add packages to ...

Policy permissions

You must have Manage Software Policy (Read) permission to reference a software package in an application component or attach it to a tier.

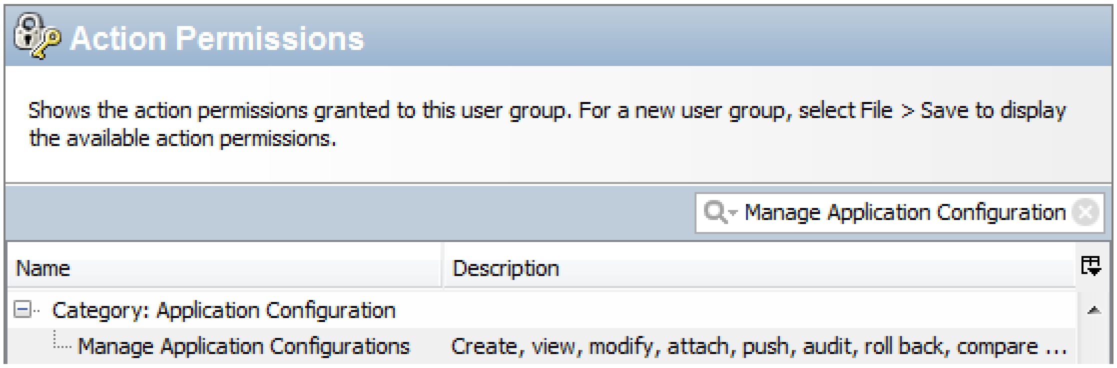


Shows the action permissions granted to this user group. For a new user group, select File > Save to display the available action permissions.

Name	Description
Category: Policy Management	
Manage Software Policy	Create, view, modify, delete, audit, remediate and attach software ...

Manage application configuration permissions

You must have Manage Application Configurations (Read) permission in order to reference an application configuration in a component.

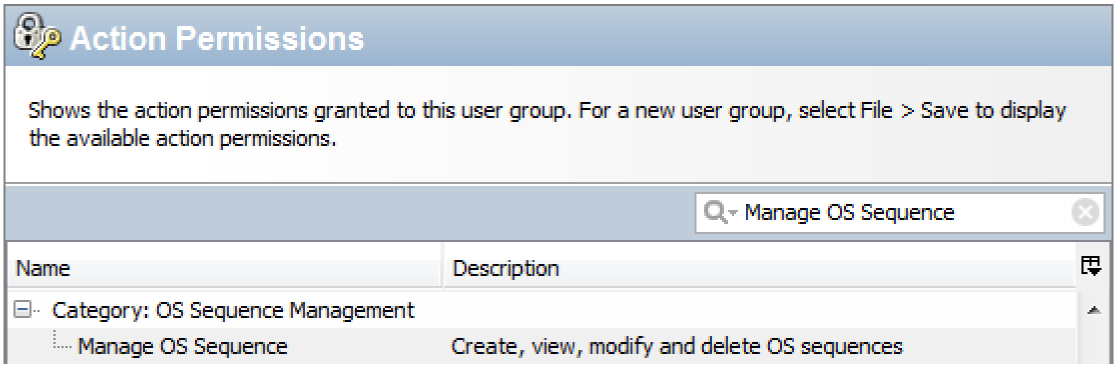


Shows the action permissions granted to this user group. For a new user group, select File > Save to display the available action permissions.

Name	Description
Category: Application Configuration	
Manage Application Configurations	Create, view, modify, attach, push, audit, roll back, compare ...

Manage OS sequence permissions

You must have Manage OS Sequence (Read) permission in order to reference an application configuration in a component.



Action Permissions

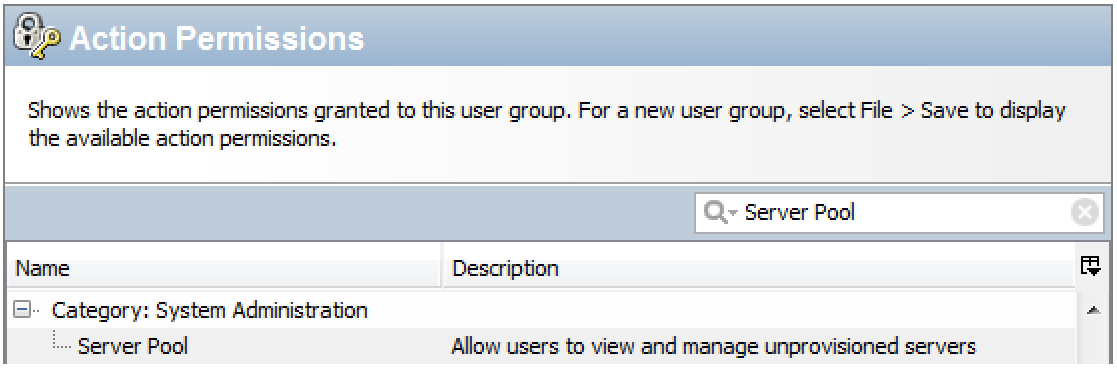
Shows the action permissions granted to this user group. For a new user group, select File > Save to display the available action permissions.

Search: Manage OS Sequence

Name	Description
⊟ Category: OS Sequence Management	
Manage OS Sequence	Create, view, modify and delete OS sequences

Server pool permissions

If you want to provision servers at deployment time by using an OS Sequence, you must have Server Pool permission.



Action Permissions

Shows the action permissions granted to this user group. For a new user group, select File > Save to display the available action permissions.

Search: Server Pool

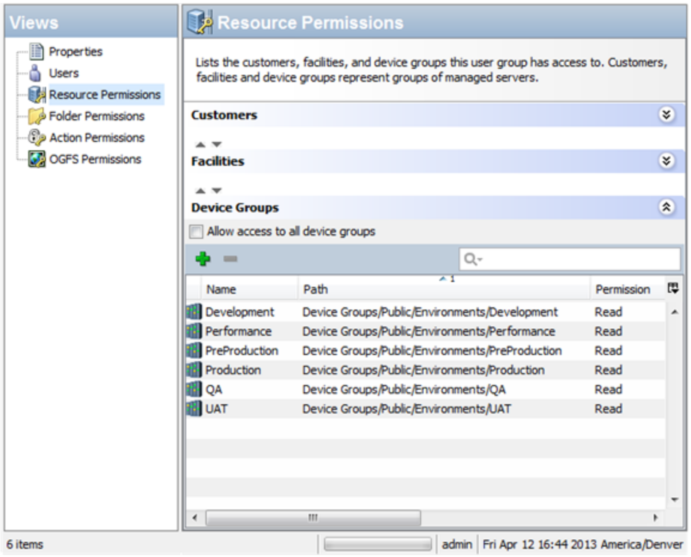
Name	Description
⊟ Category: System Administration	
Server Pool	Allow users to view and manage unprovisioned servers

For more information, see "Provisioning servers at deployment time" on page 529.

Device group permissions

If you need to create or modify Application Deployment targets, you must have Read access to the device groups that contain the servers that will be used in those targets. A QA Owner, for example, would require Read access to the QA Servers device group.

Device Group permissions are granted to user groups by the SA system administrator using the Administration screen in the SA client (for more information, see the SA 10.50 Administration Guide).



Lists the customers, facilities, and device groups this user group has access to. Customers, facilities and device groups represent groups of managed servers.

□ Allow access to all device groups

Name	Path	Permission
Development	Device Groups/Public/Environments/Development	Read
Performance	Device Groups/Public/Environments/Performance	Read
PreProduction	Device Groups/Public/Environments/PreProduction	Read
Production	Device Groups/Public/Environments/Production	Read
QA	Device Groups/Public/Environments/QA	Read
UAT	Device Groups/Public/Environments/UAT	Read

In this example, all servers in the environments listed here would be available to this user group in the Add Servers to Target dialog in the Application Deployment UI.

Library folder permissions

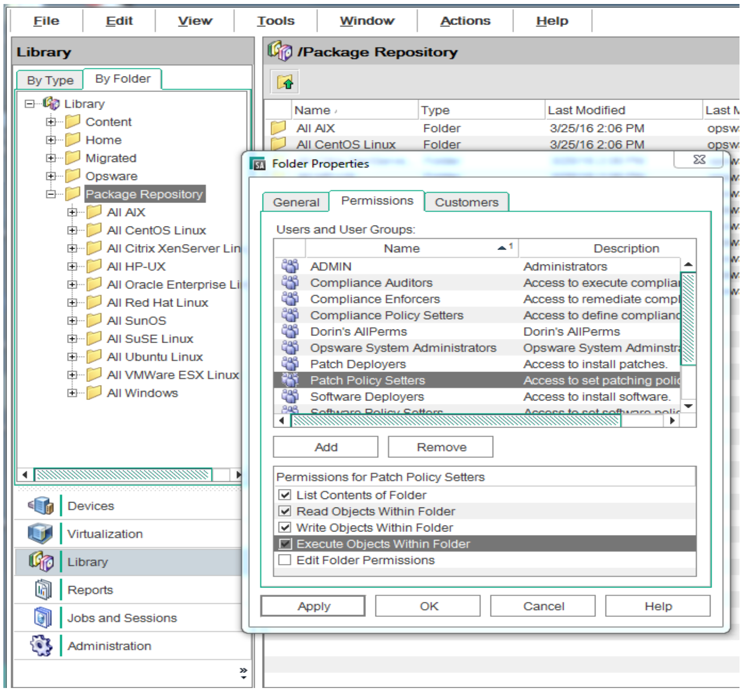
The permissions discussed in this topic are set at the Folder level on the Library screen by the SA system administrator.

If you need to create or modify applications that require the following items, you must have permission to Read and List them in the Library:

- Software Policies
- Packages
- Application Configurations
- OS Sequences

Access to items in the Library is controlled at the Folder level. "Where to set library folder permissions for packages" below shows an example of how to set these permissions for Packages.

Where to set library folder permissions for packages



Name	Type	Last Modified	Last M
All AIX	Folder	3/25/16 2:06 PM	opsw
All CentOS Linux	Folder	3/25/16 2:06 PM	opsw

Name	Description
ADMIN	Administrators
Compliance Auditors	Access to execute complia
Compliance Enforcers	Access to remediate compl
Compliance Policy Setters	Access to define complianc
Dorin's AllPerms	Dorin's AllPerms
Opsware System Administrators	Opsware System Adminstr
Patch Deployers	Access to install patches.
Patch Policy Setters	Access to set patching polic
Software Deployers	Access to install software.
Software Policy Setters	Access to set software polic

Permissions for Patch Policy Setters

- ☑ List Contents of Folder
- ☑ Read Objects Within Folder
- ☑ Write Objects Within Folder
- ☑ Execute Objects Within Folder
- □ Edit Folder Permissions

You must also have Read capability for the related Manage setting for each type of library item—for example, Manage Package (see "Action permissions" on page 584).

Application Deployment permissions

There are two types of permissions that determine what you can do in the Application Deployment UI:

- "Application permissions" below
- "Environment permissions" on page 591

Application permissions are set by the application owner—or by any user who has been granted Write permission for that application. The Application Deployment administrator can also set application permissions.

Environment permissions are set by the Application Deployment administrator.

Application permissions

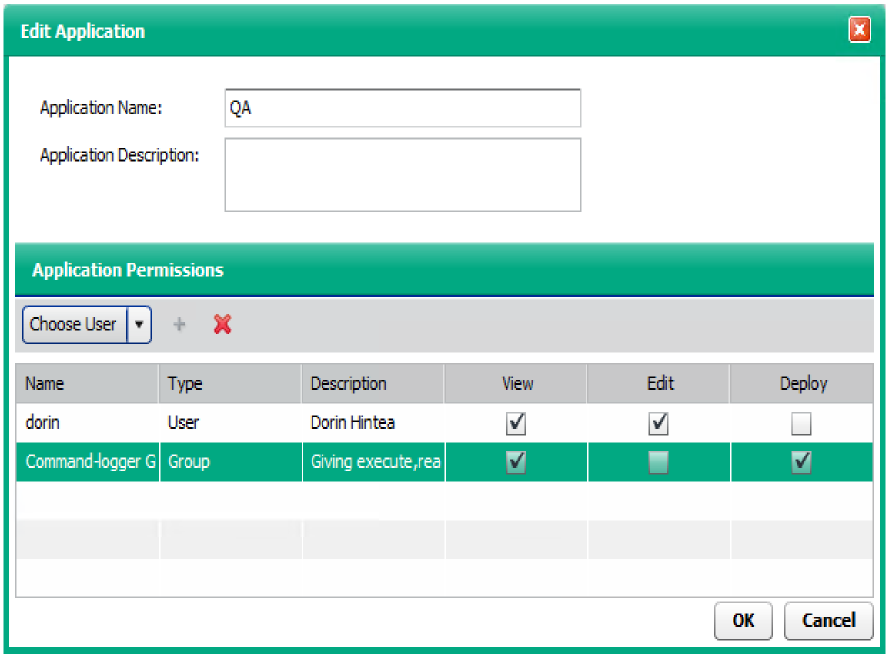
There are three types of application permissions:

- View – Determines whether the application is visible in the Manage Applications tool and the Select Release drop-down list.
- Edit – Determines who can edit the properties of the application.
- Deploy – Determines who can deploy the application. Permission to deploy to the pertinent environment is also required (see "Environment permissions" on the next page).

By default, the only users who have permission to View or Edit an application are the user who created it (the owner) and the Application Deployment administrator.

If you want other users to be able to View, Edit, or Deploy your application, you must explicitly grant them permission to do so.

Where to set application permissions



Application Name: QA

Application Description:

Name	Type	Description	View	Edit	Deploy
dorin	User	Dorin Hintea	☑	☑	☐
Command-logger G	Group	Giving execute,rea	☑	☐	☑

The users and groups available in the Choose User drop-down list are the same as the users and groups that are maintained in the SA client.

Only the application owner, a user who has been explicitly granted Edit permission for that application, and the Application Deployment administrator can modify application permissions.

To grant permission to access applications:

1. Go to the Application screen (click Applications in the lower left corner).
2. Click the icon to open the Manage Applications tool.

3. In the Manage Applications tool, locate and select the application that you want to work with.
4. Click **Edit Properties** (or double-click the application that you want to edit). The Edit Application dialog opens.
5. In the Application Permissions section, select an individual user or user group from the Choose User list.
6. Click **Add User**.
7. Select the View, Edit, or Deploy permissions that you want to grant.
8. Repeat step 5 through step 7 for each user or group to whom you want to grant permissions.
9. *Optional*: If you want to remove a user or group from the list, select that user or group in the list, and click ✖.
10. Click **OK** to close the Edit Application dialog and save your changes.

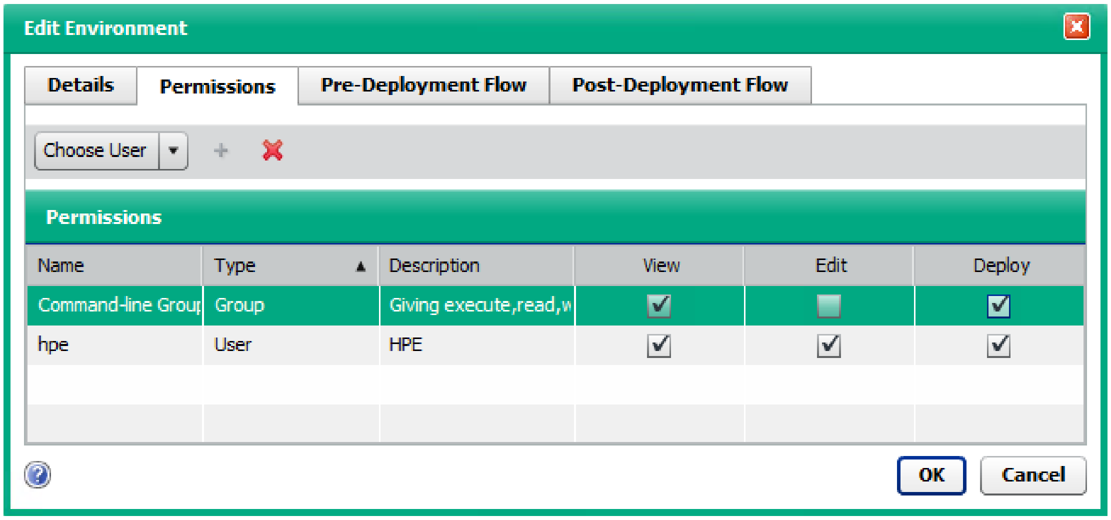
Environment permissions

There are three types of environment permissions:

- View – Determines who can see the structure of a target. You will only see targets in the Manage Targets tool and the Select Target drop-down list when you have View permission for the pertinent environment.
- Edit – Determines who can modify targets in the environment. This includes:
  - Creating targets, renaming, or deleting targets
  - Add servers to targets
  - Removing servers from targets
- Deploy – Determines who can deploy applications to targets in this environment. Permission to deploy to the pertinent application is also required (see "Application permissions" on page 589).

By default, only the Application Deployment administrator has permission to View, Edit, or Deploy to any environment. If other users need to View, Edit, or Deploy to a specific environment, the Application Deployment administrator must explicitly grant them permission to do so.

Where to set environment permissions



Name	Type	Description	View	Edit	Deploy
Command-line Grou	Group	Giving execute,read,w	☑	□	☑
hpe	User	HPE	☑	☑	☑

The users and groups available in the Choose User drop-down list are the same as the users and groups that are maintained in the SA client.

Only the Application Deployment administrator can modify environment permissions.

To grant permission to access environments:

1. Go to the Administration screen (click Administration in the lower left corner).
2. In the left panel, click Environments.
3. In the right panel, select the environment that you want to work with.
4. Click **Edit Properties** (or double-click the environment that you want to edit). The Edit Environment dialog opens.
5. Under Environment Permissions, select an individual user or user group from the Choose User list.
6. Click **Add User**. .
7. Select the View, Edit, or Deploy permissions that you want to grant.
8. Repeat step 5 through step 7 for each user or group to whom you want to grant permissions.
9. *Optional*: If you want to remove a user or group from the list, select that user or group in the list, and click ✖.
10. Click **OK** to close the Edit Environment dialog and save your changes.

Administering Application Deployment

This section shows you how to perform the functions available on the Administration screen. It includes the following topics:

- "Administering tiers" below
- "Administering environments" on page 596
- "Administering lifecycles" on page 599
- "Administering scripts" on page 601
- "Administering code component source types" on page 603
- "Administering application settings" on page 608

You must have Manage Application Deployment permission in order to see the Administration screen in the Application Deployment user interface and modify the settings discussed in this section (see "Overview" on page 579).

Administering tiers

As the Application Deployment administrator, you can manage the tiers available for creating and deploying applications. You can do the following:

- "Creating a new tier" below
- "Creating a new tier group" on the next page
- "Modifying the tier hierarchy" on page 595
- "Deleting a tier or tier group" on page 595

Each of these processes is described briefly below—all topics assume that you have Application Deployment administrator privileges.

Creating a new tier

You must have permission to Manage Application Deployment in order to administer tiers.

You can create a new tier and place it anywhere in the existing tier hierarchy. Once a tier has been created, any user with Edit permission for an application and an environment can use that tier.

To create a new tier:

1. Go to the Administration screen (click Administration in the lower left corner).
2. In the left panel, click **Tiers**.
3. In the right panel, click **Create Tier**. The New Tier dialog opens.
4. Specify the following properties:
   - Name – enter a unique name (see Naming rules).
   - Group – locate and select the group in the tier hierarchy where you want to create the new tier.
   - Platform – select Windows or UNIX.
   - Backup Directory – optionally specify the location where components in this tier will store their backup files on the target server in the event that a rollback is required.
   - *Optional:* Associated Policy – click **Select Policy**, and select a software policy from the Library. This ensures that the middleware required to support the function of this tier is in place on the target servers.
   - Deploy Behavior – specify what Application Deployment should do if the associated policy is not yet attached to a server at deployment time.
   - *Optional:* OS Provisioning – click **Select OS Sequence**, and select an OS Sequence from the Library. This will be used to install and configure the appropriate operating system on any unprovisioned servers included in the target. See "Provisioning servers at deployment time" on page 529 for more information.

   You can also modify these properties for any existing tier
5. Click **OK**.

Creating a new tier group

You must have permission to Manage Application Deployment in order to administer tier groups (see "Setting permissions" on page 579).

You can create a new tier group in the tier hierarchy. You can then move existing tiers into the new tier group or create new tiers for it.

To create a new tier group:

1. Go to the Administration screen (click Administration in the lower left corner).
2. In the left panel, click **Tiers**.
3. In the right panel, click **Create Tier Group**. The New Tier Group dialog opens.
4. Specify the following:
   - Name – enter a unique name (see Naming rules).
   - Group – locate and select the “parent” group in the tier hierarchy under which you want to create the new tier group.
5. Click **OK**.

Modifying the tier hierarchy

You must have permission to Manage Application Deployment in order to administer tiers (see "Setting permissions" on page 579).

You can move an existing tier or a tier group into any other existing tier group.

To move a tier into a different tier group:

1. Go to the Administration screen (click Administration in the lower left corner).
2. In the left panel, click **Tiers**.
3. In the right panel, double-click any tier or tier group.
4. Locate and select the “parent” group in the tier hierarchy where you want to move the selected item.
5. Click **OK**.

Deleting a tier or tier group

You must have permission to Manage Application Deployment in order to administer tiers (see "Setting permissions" on page 579).

You can delete any existing tier or tier group that is not In Use (used in a version that is currently deployed to a target server or part of a release or target). If any tiers in a tier group are In Use, you cannot delete the tier group.

To delete a tier or tier group:

1. Go to the Administration screen (click Administration in the lower left corner).
2. In the left panel, click **Tiers**.
3. In the right panel, select the tier or tier group that you want to delete.
4. Click **Delete**.
5. Click **Yes** to confirm.

If you want to delete a tier that is In Use, you must first do the following:

- Undeploy and delete all versions that use the tier.
- Delete any releases that use the tier.
- Delete any targets that use the tier.
- Only empty tier groups can be deleted.

It is very difficult to delete a tier after it has been used.

Administering environments

As the Application Deployment administrator, you can manage the available environments for application deployment. You can do the following:

- "Creating a new environment" below
- "Changing permissions for an environment" on page 598
- "Specifying OO flows for an environment" on page 598
- "Deleting an environment" on page 599

Each of these processes is described briefly below—all topics assume that you have Application Deployment administrator privileges.

Creating a new environment

You must have permission to Manage Application Deployment in order to administer environments (see "Setting permissions" on page 579).

You can create a new environment and make it available to specific users or user groups. You can specify an OO flow to initiate prior to each deployment and another to initiate after each deployment.

Environments in the Application Deployment context are mirrored as SA device groups. When you create a new environment, the corresponding device group will be visible after the next synchronization occurs.

- To force an immediate synchronization, right-click an environment and select **Export**.
- To view the mirrored device group, right-click an environment, and select **View Device Group**.

To create a new environment:

1. Go to the Administration screen (click Administration in the lower left corner).
2. In the left panel, click **Environments**.
3. In the right panel, click the “Create Environment” button: . The New Environment dialog opens.
4. On the Details tab, specify the following properties:
   - Name – enter a unique name (see Naming rules).
   - Initial Status – specify the default status of the environment. This determines whether an environment is, by default, able to accept applications or if it requires explicit permission for every version. Users with permission to deploy to this environment will be able to change this status on the Deployment screen.
   - Undeploy – select this box to establish the default auto-undeploy setting. The user can override this setting.
5. On the Permissions tab, specify which users are allowed to View, Edit, and Deploy to this environment.
   - Users with View permission can see this environment in the Select Targets drop-down list (on the Targets and Deploy screens) and the Manage Targets tool (accessible from the Targets screen).
   - Users with Edit permission can open the Edit Environment dialog and change the properties of the environment. They can also create and modify targets.
   - Users with Deploy permission can deploy applications (that they have permission to deploy) to this environment.

     To specify user permissions, follow these steps:

   a. Select an individual user or user group from the Choose User list.
   b. Click the “Add User” button: +.
   c. Select the View, Edit, or Deploy permissions that you want to grant.
   d. Repeat step a through step c for each user or group to whom you want to grant permissions.

   e. *Optional*: If you want to remove a user or group from the list, select that user or group in the list, and click ✖.

6. *Optional*: On the Pre-Deployment Flow tab, select the OO flow that you want to initiate just prior to all deployments on target servers in this environment. For example, you might want to disable monitoring in the Production environment prior to deploying.

   Click the Change button, and select a flow from the OO library. Double-click the Value cell for any parameter whose value you want to change (see "Parameters and special variables" on page 502).

   You can reference a global parameter or release parameter when specifying parameter values for the Pre-Deployment or Post-Deployment OO Flow (see "Parameters and special variables" on page 502).

7. *Optional*: On the Post-Deployment Flow tab, select the OO flow that you want to initiate immediately after all deployments on target servers in this environment. For example, you might want to re-enable monitoring after a deployment.

8. Click **OK** to close the New Environment dialog and save your changes.

Changing permissions for an environment

You must have permission to Manage Application Deployment in order to administer environments (see "Setting permissions" on page 579).

You can change the View, Edit, and Deploy permissions for any environment. See step 5 under "Creating a new environment" on page 596 for instructions.

Specifying OO flows for an environment

You must have permission to Manage Application Deployment in order to administer environments (see "Setting permissions" on page 579).

You can change the Pre-Deployment and Post-Deployment Flows for any environment. See step 6 and step 7 under "Creating a new environment" on page 596 for instructions.

Deleting an environment

You must have permission to Manage Application Deployment in order to administer environments (see "Setting permissions" on page 579).

You can delete any existing environment that is not In Use (used in a version that is currently deployed to a target server or part of a lifecycle).

To delete an environment:

1. Go to the Administration screen (click Administration in the lower left corner).
2. In the left panel, click **Environments**.
3. In the right panel, select the environment that you want to delete.
4. Click **Delete**.
5. Click **Yes** to confirm.

If you want to delete an environment that is In Use, you must first undeploy all versions that use that environment. You must also remove the environment from any lifecycles to which it belongs.

Administering lifecycles

As the Application Deployment administrator, you can manage the available lifecycles for application deployment. You can do the following:

- "Creating a lifecycle" below
- "Modifying a lifecycle" on the next page
- "Deleting a lifecycle" on page 601

Each of these processes is described briefly below—all topics assume that you have Application Deployment administrator privileges.

Creating a lifecycle

You must have permission to Manage Application Deployment in order to administer lifecycles (see "Setting permissions" on page 579).

Lifecycles are used to facilitate the orderly progress of software applications from the Development environment through various levels of testing environments and, ultimately, to the Production environment. Lifecycles enable you to control the promotion of an application from one environment to the next by explicitly allowing or preventing deployment to the various environments.

You can create new lifecycles, change existing lifecycles, and delete lifecycles that are not in use.

Application Deployment lifecycles are not enforced in any way by SA. They are only meaningful within the Application Deployment context.

To create a new lifecycle:

1. Go to the Administration screen (click Administration in the lower left corner).
2. In the left panel, click **Lifecycles**.
3. In the right panel, click the “Create Lifecycle” button: . The New Lifecycle dialog opens.
4. Specify the following properties:
   - Name – enter a unique name (see Naming Rules).
   - Lifecycle – use the horizontal arrows to move one or more Available Environments into the Lifecycle box; use the vertical arrows to modify their order in the Lifecycle.
5. Click **OK** to close the New Lifecycle dialog and save your changes.

Modifying a lifecycle

You must have permission to Manage Application Deployment in order to administer lifecycles (see "Setting permissions" on page 579).

You can rename or re-arrange the environments in a lifecycle.

To edit a lifecycle:

1. Go to the Administration screen (click Administration in the lower left corner).
2. In the left panel, click **Lifecycles**.
3. In the right panel, select the lifecycle that you want to edit.
4. Click **Edit Properties**. The Edit Lifecycle dialog opens.
5. *Optional*: Enter a new name for the lifecycle.
6. Optional: Use the horizontal arrows to move one or more Available Environments into the Lifecycle box; use the vertical arrows to modify their order in the Lifecycle.
7. Click **OK** to close the Edit Lifecycle dialog and save your changes.

Deleting a lifecycle

You must have permission to Manage Application Deployment in order to administer lifecycles (see "Setting permissions" on page 579).

You can delete any existing lifecycle that is not In Use (used in a version or release that is currently deployed to a target server).

To delete a lifecycle:

1. Go to the Administration screen (click Administration in the lower left corner).
2. In the left panel, click **Lifecycles**.
3. In the right panel, select the lifecycle that you want to delete.
4. Click **Delete**.
5. Click **Yes** to confirm.

To delete a lifecycle, it must not be associated with any release or version.

Administering scripts

As the Application Deployment administrator, you can manage the available scripts used for deploy, backup, rollback, and undeploy operations. These scripts are referenced by Code, Configuration File, and Windows Registry components. You can do the following:

- "Creating a new script" on the next page
- "Modifying an existing script" on the next page
- "Deleting a script" on page 603

Each of these processes is described briefly below—all topics assume that you have Application Deployment administrator privileges.

The contents of the script are copied when a version is created. Changes to scripts do not affect versions created previously.

Creating a new script

You must have permission to Manage Application Deployment in order to administer scripts.

Scripts are referenced in application components. They are used to facilitate deployment, backup, rollback, and undeploy operations. A set of basic scripts is provided “out of the box.” You can modify these scripts, or you can create new scripts. Users who have permission to create applications can reference these scripts when they create components.

To create a new script:

1. Go to the Administration screen (click Administration in the lower left corner).
2. In the left panel, click **Scripts**.
3. In the right panel, click **Create Scrip**t. The New Script dialog opens.
4. Specify the following properties:
   - Name – enter a unique name (see "Naming rules" on page 438).
   - Component – type of component that will reference this script.
   - Purpose – select Deploy, Undeploy, Backup, or Rollback. Different components use scripts in different deployment phases. Only Windows Registry components, for example, use Deploy scripts.
   - Platform – select Windows or UNIX.
   - Primary – select this box if there are multiple scripts that serve this purpose for this type of component on this platform, and this is the default script for new components to use. This is useful, for example, when you are developing or testing a new script.
   - Type – select the script type. Available choices depend on the platform specified.
   - Content – enter the actual instructions that constitute the script.
5. You can also modify these properties for any existing script.
6. Click **OK** to save your changes and close the New Script dialog.

Modifying an existing script

You must have permission to Manage Application Deployment in order to administer scripts (see "Setting permissions" on page 579).

You can modify any existing script by using the Edit Script dialog.

To modify a script:

1. Go to the Administration screen (click Administration in the lower left corner).
2. In the left panel, click **Scripts**.
3. In the right panel, select the script that you want to modify.
4. Click **Edit Properties**.. The Edit Script dialog opens.
5. Make any changes to the script’s properties (see "Creating a new script" on the previous page).
6. Click **OK** to save your changes and close the Edit Script dialog.

Deleting a script

You must have permission to Manage Application Deployment in order to administer scripts (see "Setting permissions" on page 579).

You can delete any existing script that is not In Use (used in a release).

To delete a script:

1. Go to the Administration screen (click Administration in the lower left corner).
2. In the left panel, click **Scripts**.
3. In the right panel, select the script that you want to delete.
4. Click **Delete**.
5. Click **Yes** to confirm.

If you want to delete a script, it must not be referenced by any release.

Administering code component source types

Application developers use Code components to specify files required by their applications. Code component source types tell Application Deployment how to access those files.

Source types can be enabled or disabled. At least one source type must be enabled at all times. Source types that are referenced cannot be disabled. Before you can disable a source type, you must delete any versions that reference that source type and modify any releases that reference that source type.

Application Deployment uses CruiseControl, an open source build system, to manage Code components whose source type is Filesystem or a specific source code control system (such as CVS or Subversion). For these source types, you must provide an XML snippet that contains the required CruiseControl configuration settings. Examples are provided here for a "Filesystem" below and a "Source code control system" on the next page.

For more information about configuring CruiseControl, see the following resource:

http://cruisecontrol.sourceforge.net/main/configxml.html

Editing a code component source type

You must have permission to Manage Application Deployment in order to administer code component source types.

The Edit Code Component Source Type dialog enables you to configure the settings for any type of Code component source. Regardless of what type of Code component source you are editing, you must begin by opening this dialog.

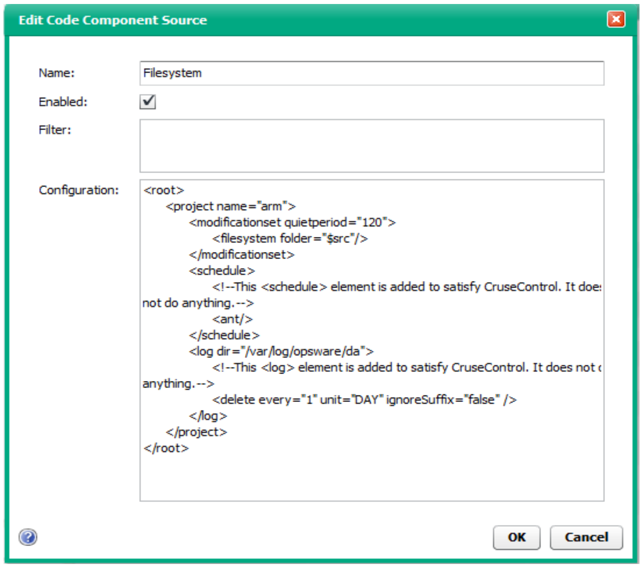
To edit a code component source:

1. Go to the Administration screen (click **Administration** in the lower left corner).
2. In the left pane, select **Code Component Source Types**.
3. In the right pane, do one of the following things:
   - To create a new source type, click **Create Source Type**.
   - To modify an existing source type, select it, and click **Edit Properties**. The Edit Lifecycle dialog opens.
4. To make this Code component source type available to application developers, select **Enabled.**
5. Follow the detailed instructions below for the specific component source type that you are editing:
   - Filesytem
   - URL
   - Source Code Control System

Filesystem

To configure the Filesystem source for a Code component, you must specify the CruiseControl settings shown here.

Settings for the filesystem source type



Name: Filesystem

Enabled: ☑

Filter:

Configuration:

```
<root>
    <project name="arm">
        <modificationset quietperiod="120">
            <filesystem folder="$src"/>
        </modificationset>
        <schedule>
            <!--This <schedule> element is added to satisfy CruseControl. It doe
not do anything.-->
            <ant/>
        </schedule>
        <log dir="/var/log/opsware/da">
            <!--This <log> element is added to satisfy CruseControl. It does not
anything.-->
            <delete every="1" unit="DAY" ignoreSuffix="false" />
        </log>
    </project>
</root>
```

The `quietperiod` is the number of seconds required to have passed since the last modification occurred before a new version can be created.

The `$src` variable represents the Source Directory specified in the Code component.

URL

The only configurable settings for a URL Code component source are its Name and the Enable field. The application developer must specify the URL itself. See "Using a URL" on page 474.

Source code control system

To integrate with source code control systems (such as CVS or Subversion), Application Deployment embeds CruiseControl, an open source build system. The CruiseControl configuration is stored as part of the Source Type as an XML snippet. Other source code control systems can be added by simply adding new XML snippets.

The time on your Server Automation cores must be synchronized with your source code control system servers. Otherwise, the determination of which files are included in a Code component can be wrong due to the time skew.

Be sure to install the source code control system client software on the SA Core server before you attempt to integrate it with Application Deployment.

To integrate a source code control system:

Install the source code control system client software on the SA Core server.

On the SA Core server, create a client copy of the source that you want to include in a Code component. In CVS and Subversion, this is called “checking out” a repository.

Open the Edit Code Component Source dialog (see "Editing a code component source type" on page 604 above).

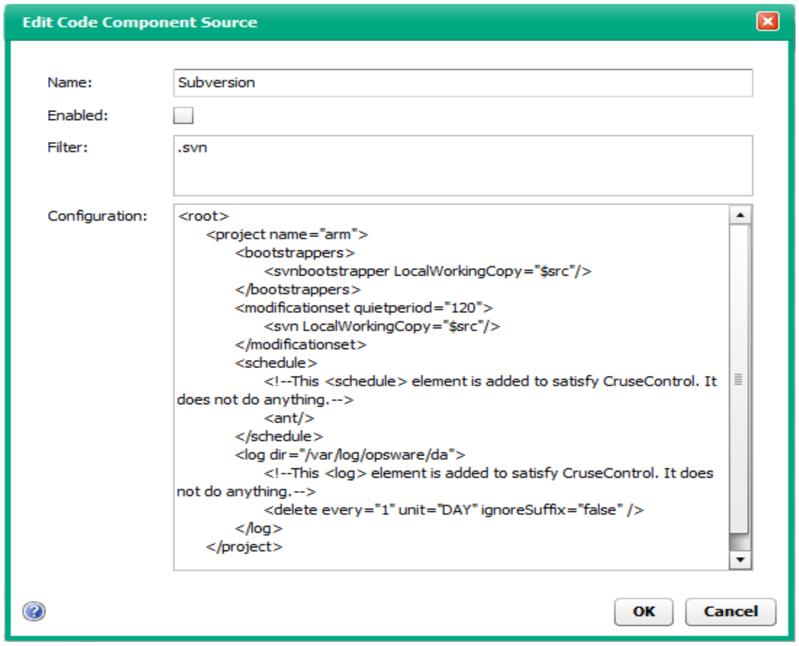
Add the CruiseControl configuration and any filter criteria required for your source code control system.

After you have done this, your application developers can reference the path to this client copy in their Code components. See "Using a source code control System" on page 473.

Example: Subversion

The following is a sample configuration for a Subversion Code component source.

Settings for the subversion source type



Name: Subversion

Enabled: □

Filter: .svn

Configuration:

```
<root>
    <project name="arm">
        <bootstrappers>
            <svnbootstrapper LocalWorkingCopy="$src"/>
        </bootstrappers>
        <modificationset quietperiod="120">
            <svn LocalWorkingCopy="$src"/>
        </modificationset>
        <schedule>
            <!--This <schedule> element is added to satisfy CruseControl. It
does not do anything.-->
            <ant/>
        </schedule>
        <log dir="/var/log/opsware/da">
            <!--This <log> element is added to satisfy CruseControl. It does
not do anything.-->
            <delete every="1" unit="DAY" ignoreSuffix="false" />
        </log>
    </project>
```

To check out this Subversion module on the SA Core server, you would use the following command:

```
svn checkout URL
```

For example:

```
svn checkout https://mySVNserver.mycompany.com:444/svn/proj1/trunk
```

Setting up an SVN client in an SA multi-core or multi-slice environment

To avoid having to install a Subversion (SVN) client on all the servers in a multi-core or multi-slice environment, you can use a Network File System (NFS) mount. The following process has been tested on Linux servers:

On the master core server:

1. Install the SVN client—for example, the CollabNet Subversion client—in the following directory: `/opt/CollabNet_Subversion`
2. Add the following line to the `/etc/exports` file: `/opt/CollabNet_Subversion *(ro)`
3. Restart the NFS service using the following command: `/etc/init.d/nfs restart`

On the second server (and subsequent servers):

1. Mount to the exported drive of the Master core. For example, perform the following steps:

   ```
   cd /opt
   mkdir CollabNet_Subversion
   mount <MasterCore>:/opt/CollabNet_Subversion /opt/CollabNet_Subversion
   ```

   Here, *MasterCore* can be either the DNS Name or IP address of the Master core server.

2. Create a link to SVN from `/usr/bin/svn`:

   ```
   ln -s /opt/CollabNet_Subversion/bin/svn /usr/bin/svn
   ```

You can now use SVN as if it were installed on the second server.

You can use the same NFS mount strategy to set up a CVS client.

Administering application settings

You can manage the application settings that determine how Application Deployment functions.

To modify the application settings:

1. Go to the Administration screen (click **Administration** in the lower left corner).
2. In the left pane, select **Application Settings**.
3. Modify one or more of the following settings:
   - "Version numbering" below
   - "Just-in-time targets flow" on the next page
   - "Code and configuration file component base directories" on the next page
   - "Global parameters" on page 612
4. Click **Save** to save your changes

You must have permission to Manage Application Deployment to modify these settings.

Version numbering

This group contains the following three settings:

Initial version

Specifies how the initial version of a release should be named or numbered. The default is 1. If you specify a number, Application Deployment will attempt to increment that number each time that it creates a new version.

The value that you specify here will appear in the Version box in the Create New Version dialog whenever a user creates the very first version of the release. The user can override the initial value.

Version context

Specifies how the version label should be constructed.

Select Use Release Name if you want to concatenate the release name with the Initial Version number to form the version label.

Software deploy method

Select the default method for deployment of Code and Package components.

Whenever you create a new version, Application Deployment creates a package that contains the files or package specified and any scripts required. In addition, you can specify one of the following options:

- Attach Policy as Default – Also create a policy when a new version is created; when this version is deployed, attach and remediate this policy on each target server.
- Install Package as Default – Perform an “ad hoc” install of the package on each target server.
- Always Attach Policy – Always create a policy; attach, and remediate.
- Always Install Package – Always perform an “ad hoc” install of the package.

If you select one of the “as Default” options, the user can override the Deploy Method when creating or editing components (see "Components" on page 468).

If you select one of the “Always” options, the “Locked” indicator is displayed, and the user cannot override the Deploy Method.

Just-in-time targets flow

This setting specifies the Operations Orchestration (OO) flow that Application Deployment will use to create just-in-time targets at deployment time (see "Just-in-time targets" on page 526).

This flow must exist in your OO library. For more information, see "Finding and selecting an item in the library" on page 500.

This setting is only available when SA is configured to integrate with OO.

Code and configuration file component base directories

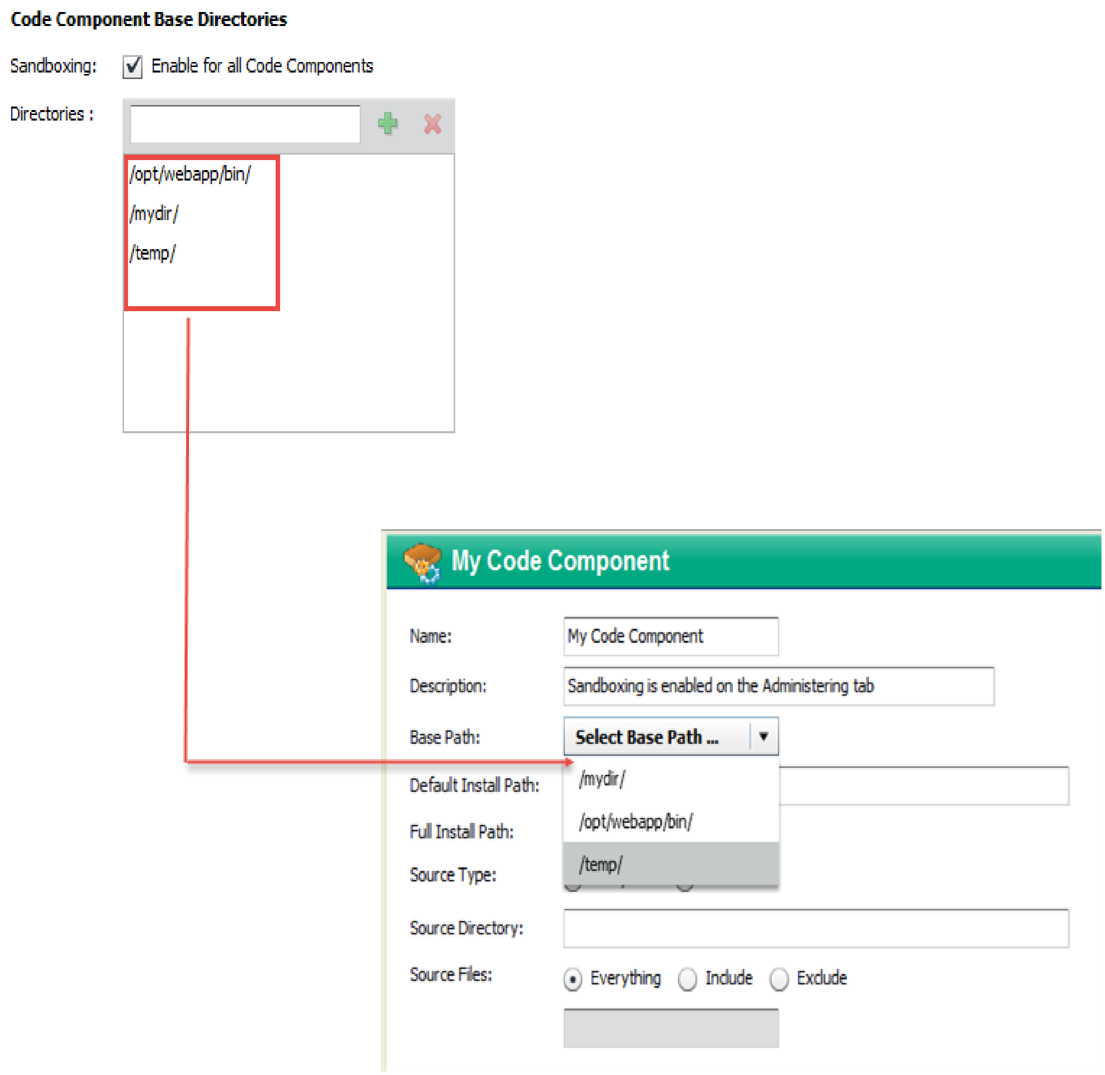
These settings enable you to create separate “sandboxes” for Code and Configuration File component deployment. This is useful when you want files created by these components to only be installed in controlled file system locations.

The following example describes how the sandboxing mechanism works for a Code component. It works the same way for Configuration File components.

How it works

When sandboxing is enabled, you can specify a list base directories (sandboxes) for Code components and Configuration File components. These directories then appear in the Base Path drop-down list in the component properties editor on the Applications tab.

Example of base directories



Code Component Base Directories

Sandboxing: ☑ Enable for all Code Components

Directories :

/opt/webapp/bin/
/mydir/
/temp/

My Code Component

Name: My Code Component

Description: Sandboxing is enabled on the Administering tab

Base Path: Select Base Path ...

/mydir/
/opt/webapp/bin/
/temp/

Default Install Path:

Full Install Path:

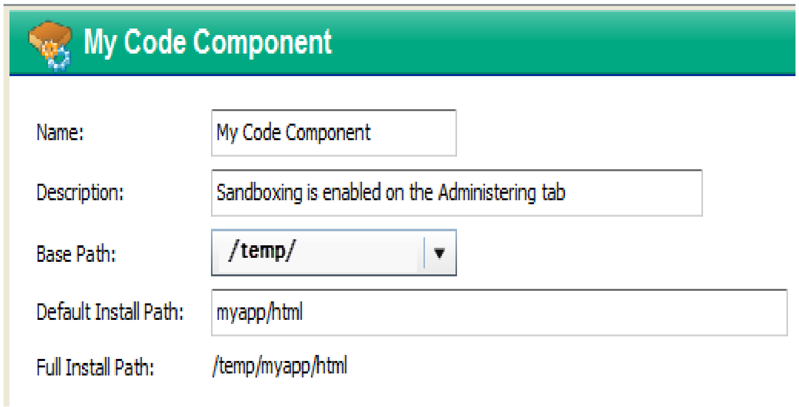
Source Type:

Source Directory:

Source Files: ● Everything ○ Include ○ Exclude

In the Code component, the selected Base Path is combined with the specified Default Install Path to form the Full Install Path—where the files will be placed on the target.

Code component with sandboxing



My Code Component

Name: My Code Component

Description: Sandboxing is enabled on the Administering tab

Base Path: /temp/

Default Install Path: myapp/html

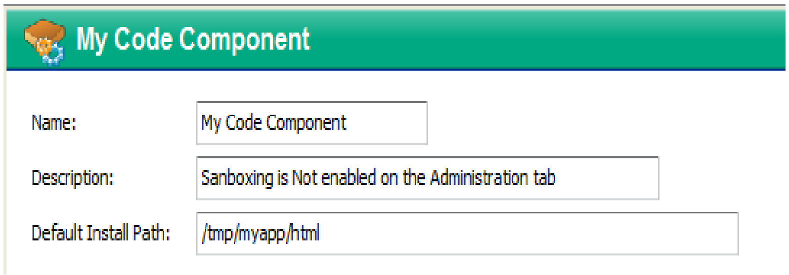
Full Install Path: /temp/myapp/html

When sandboxing is enabled, the application designer must specify a Base Path when creating a Code or Configuration File component. A version cannot be created for release that contains a Code or Configuration File component whose Base Path has not been specified (when sandboxing is enabled).

If you remove a Base Directory that is used by a component in an existing release, new versions of that release cannot be created until a new Base Path is specified in the component. Existing versions, however, can still be deployed.

When sandboxing is not enabled, the Base Path and Full Install Path fields do not appear in the Code component properties editor.

Code components without sandboxing



My Code Component

Name: My Code Component

Description: Sanboxing is Not enabled on the Administration tab

Default Install Path: /tmp/myapp/html

How to configure

You can configure sandboxing for Code components, Configuration File components, or both.

To configure sandboxing:

1. In the left pane of the Administration screen, select **Application Settings**.
2. In the right pane, select one or both of the following options:
   - **Enable for all Code Components**
   - **Enable for all Configuration File Components**

3. For each component type that you want to sandbox, follow these steps:
   a. In the “Directories” text box, type the base path directory. Only absolute paths are permitted. Relative paths (such as `../webapps`) are not allowed.
   b. Click **Add directory choice**.
   c. Repeat steps (a) and (b) for each base directory that you want to add.
4. Click **Save** to save your changes

To remove an existing base path directory:

1. In the Directories box, select an existing directory.
2. Click **Remove directory choice**.
3. Click **Save** to save your changes.

You can only modify the Directories list when sandboxing is enabled.

Global parameters

Global parameters are available to all applications. SA users who have permission to create applications or edit a specific application can reference global parameters when specifying the default value of a component parameter or the value of a release parameter (Parameters and Special variables see "Parameters and special variables" on page 502).

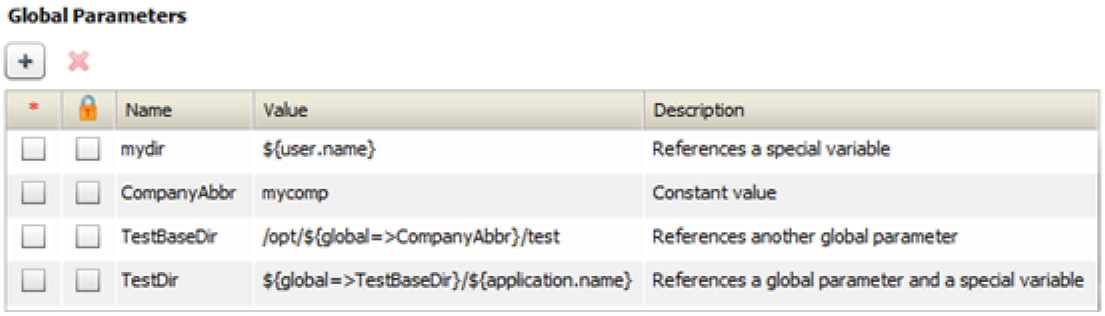
SA users who have Deploy permission for an application and Write permission for the pertinent environment can use global parameters to override the default parameter values at deployment time by following similar steps in the Parameter Editor (see "Customizing deployment parameters" on page 544).

To define or modify a global parameter:

1. On the Application Settings page, scroll down to the Global Parameters table.
2. To add a new global parameter, follow these steps:
   a. Click **Add New Global Parameter**.
   b. Enter the Name of your new global parameter. The name must be unique among the global parameters. The name cannot include any of these characters: @, \, {, or }.
   c. Click **OK**.
3. Click anywhere in the Value column for this parameter.

4. Specify the value of the parameter (for detailed instructions, see "Specifying the value of a parameter" on page 511).

   You can reference special variables and other global parameters in the value specification. For example:



Global Parameters

*	🔒	Name	Value	Description
□	□	mydir	${user.name}	References a special variable
□	□	CompanyAbbr	mycomp	Constant value
□	□	TestBaseDir	/opt/${global=>CompanyAbbr}/test	References another global parameter
□	□	TestDir	${global=>TestBaseDir}/${application.name}	References a global parameter and a special variable

5. *Optional:* To encrypt the parameter value, select the box in the 🔒 (encrypted) column.

   Note that the * (required) column is not pertinent in this context.

6. Click **Save** on the tool bar to save your changes.

You can also delete any global parameter that is not currently referenced by any other parameter.

To delete a global parameter:

1. In the Global Parameters table, select the parameter that you want to delete.
2. Click **Delete Global Parameter**. A confirmation dialog appears.
3. Click **Yes** to confirm the delete.
4. Click **Save** on the tool bar to save your changes.

Importing and exporting data

This topic contains instructions for importing and exporting Application Deployment data from Server Automation (SA). It contains the following topics:

- "Working" on the next page
- "Special considerations" on the next page

Working

You can export Application Deployment data to an XML file or import data from an XML file. This is useful when you want to copy or move your Application Deployment data from one SA core to another, back up your data for archival purposes, or create an Application Deployment scenario using scripts.

The following items are imported and exported:

- Lifecycles
- Environments
- Applications
- Application Groups
- Tiers and Tier Groups
- Releases
- Components
- Scripts
- Source Types

You can preview the results of a potential import to see what would be created and determine whether any conflicts (collisions) would occur with existing Application Deployment data.

You can import or export the entire Application Deployment database, or you can import or export specific applications. When you import or export a specific application, all items associated with that application—releases, lifecycles, environments, source types, components (including rollback and undeploy scripts, if applicable), and tiers—are included.

Special considerations

The following items are not exported:

- Release versions
- Deployment jobs
- Targets or target groups
- Environment pre- or post-deployment flows

- Application Settings (on the Administration screen)
- Application permissions
- Environment permissions
- SA library objects (Software Policies, Application Configurations, and Packages)
- HPE OO flow content or definitions
- These items also cannot be imported.

Because you cannot import or export release versions, there is no way to import or export delta release information.

For an import operation, if an SA library item does not exist on the SA Core where the import is performed, that item will not be visible in the object that contains it. For example, if a tier has a policy, and that policy does not exist on that SA Core, the policy will be blank.

You can specify what happens if a conflict is encountered during an import operation: skip the item or overwrite it. The default behavior is to skip the item and continue the import. If you specify `--importConflict overwrite`, however, conflicting items are updated based on the contents of the import XML file.

Command Line tool

Application Deployment provides a command line tool to help you import and export application deployment data. It has the following syntax:

`/opt/opsware/da/bin/admtool.sh` -*opType* `[--`*appListTypeappList*|*appListFile*`] --` *fileTypefilePath* `[--importConflict overwrite|skip]`

You can use this tool to import or export all the applications or a subset thereof.

Options for admtool.sh

Option	Description
opType	Specifies the operation: `import`, `export`, or `preview`. You can abbreviate this option by specifying `-i`, `-e`, or `-p`.
appListType	Optional: Indicates that specific applications will be imported or exported and how the application names will be specified: `importAppList` – application names specified on the command line `exportAppList` – application names specified on the command line

Options for admtool.sh, continued

Option	Description
	`importAppListFile` - application names specified in a CSV file `exportAppListFile` - applications names specified in a CSV file If you do not specify this option, all applications are imported or exported.
appList	List of applications that will be imported or exported. Application names must be separated with semicolons: `app1;app2;app3;...appN` Used only with the `importAppList` and `exportAppList` options.
appListFile	CSV file containing the list of applications that will be imported or exported. Application names must be separated with semicolons: `app1;app2;app3;...appN` Used only with the `importAppListFile` and `exportAppListFile` options.
fileType	Specify `importFile` for import or preview operations. Specify `exportFile` for export operations.
filePath	Full path and name of the XML file to import to, export from, or preview. If you do not specify a file name for an export operation, a default export file is used. You can specify the default export file in the `/etc/opt/opsware/da/da.conf` file as follows: `exportFile=/var/tmp/my_export_default_file` If `exportFile` is not specified in the `da.conf` file, and no export file is specified on the command line, the exported data is written to the following file: `/opt/opsware/da/bin/ADMExport.xml`
`importConflict`	Optional: Indicates what happens if a conflict occurs during an import operation: `skip` or `overwrite`. A conflict occurs if any item in the import XML file already exists in the Application Deployment database. `skip` leaves any item with a conflict unchanged in the database. This is the default behavior. `overwrite` updates any item with a conflict based on the contents of the import file (see "Overwrite conflict resolution" on page 618). You can abbreviate this option by specifying `-c overwrite` or `-c skip`.

Syntax examples

The following tables show the correct syntax for each type of import and export scenario that the Application Deployment Manager supports.

Preview syntax examples

Preview scenario	Example
Full Import Preview (all applications in the specified XML file)	`/opt/opsware/da/bin/admtool.sh -p --importFile myImportData.xml`
Full Import Preview with Overwrite on Conflict	`/opt/opsware/da/bin/admtool.sh -p --importFile myImportData.xml --importConflict overwrite`
Partial Import Preview of Specific Applications (listed on the command line)	`/opt/opsware/da/bin/admtool.sh -p --importAppList "app1;app2;app3" --importFile myImportData.xml`
Partial Import Preview of Specific Applications (listed on the command line) with Overwrite on Conflict	`/opt/opsware/da/bin/admtool.sh -p --importAppList "app1;app2;app3" --importFile myImportData.xml --importConflict overwrite`
Partial Import Preview of Specific Applications (listed in a CSV file)	`/opt/opsware/da/bin/admtool.sh -p --importAppListFile app_list.csv --importFile myImportData.xml`
Partial Import Preview of Specific Applications (listed in a CSV file) with Overwrite on Conflict	`/opt/opsware/da/bin/admtool.sh -p --importAppListFile app_list.csv --importFile myImportData.xml --importConflict overwrite`

See "Preview examples" on page 621 for additional information.

Import syntax examples

Import scenario	Example
Full Import (all applications in the specified XML file)	`/opt/opsware/da/bin/admtool.sh -i --importFile myImportData.xml`
Full Import with Overwrite on Conflict	`/opt/opsware/da/bin/admtool.sh -i --importFile myImportData.xml --importConflict overwrite`
Partial Import of Specific Applications (listed on the command line)	`/opt/opsware/da/bin/admtool.sh -i --importAppList "app1;app2;app3" --importFile myImportData.xml`
Partial Import of Specific Applications (listed on the command	`/opt/opsware/da/bin/admtool.sh -i --importAppList "app1;app2;app3" --importFile myImportData.xml --`

Import syntax examples, continued

Import scenario	Example
line) with Overwrite on Conflict	`importConflict overwrite`
Partial Import of Specific Applications (listed in a CSV file)	`/opt/opsware/da/bin/admtool.sh -i --importAppListFile app_list.csv --importFile myImportData.xml`
Partial Import of Specific Applications (listed in a CSV file) with Overwrite on Conflict	`/opt/opsware/da/bin/admtool.sh -i --importAppListFile app_list.csv --importFile myImportData.xml --importConflict overwrite`

See "Import examples" on page 623 for additional information.

Export syntax examples

Export scenario	Example
Full Export of All Application Deployment Data	`/opt/opsware/da/bin/admtool.sh -e --exportFile myExportData.xml`
Partial Export of Specific Applications (listed on the command line)	`/opt/opsware/da/bin/admtool.sh -e --exportAppList "app1;app2;app3" --exportFile myExportData.xml`
Partial Export of Specific Applications (listed in a CSV file)s	`/opt/opsware/da/bin/admtool.sh -e --exportAppListFile app_list.csv --exportFile myExportData.xml`

See "Export examples" on page 626 for additional information.

In these examples, the `app_list.csv` file contains a list of application names separated by semicolons (;). For example: `app1;app2;app3;app4`

Overwrite conflict resolution

A conflict occurs when an item in the import XML file already exists in the Application Deployment database on this SA core. By default, the conflicted item is skipped during the import.

If you specify the `--importConflict overwrite` option, however, the conflicted item is modified instead. Certain items associated with a specific release can also be deleted.

The following table summarizes the results of an import operation when `--importConflict overwrite` is specified.

Conflict resolution scenarios

Type of item	In import file not in database	In import file in database	Not in import file in database
Application Application group Release associated with a specific application Environment Lifecycle Tier Tier group Script Source type	Item is created	Item is updated based on the content of the import file	Item remains in the database and is not modified
Tier associated with a specific release Component associated with a tier that is associated with a specific release	Item is created	Item is updated based on the content of the import file	Item is deleted

See "Messages regarding data conflicts" on the next page under "Important messages" on the next page for examples of messages that pertain to import conflicts.

Log files

The `admtool.sh` tool produces a detailed log file. The file name depends on the operation performed:

```
/var/log/opsware/da/importData.log
```

```
/var/log/opsware/da/exportData.log
```

```
/var/log/opsware/da/previewData.log
```

Important messages

For a preview operation, the following type of message (on `stdout`) indicates that something new will be created if this import is performed:

```
Import: 'CodeComponentSourceType' 'QA' will be created.
```

The following type of message indicates that an application will be skipped because it was not included in the list of applications to be imported or exported:

```
Skip: Application 'AppD' due to Application Filter.
```

For an import operation, the following type of message (on `stdout`) indicates that a library item—such as a Software Policy, Package, or Application Configuration—does not exist on the destination SA core:

```
Not Found:Package'OPSWpytwist2-40.0.0.7.325.zip(SunOS 5.8)'does not exist
```

The following types of messages indicate whether a conflict exists between the existing Application Deployment data on the destination SA core and the data in the import XML file:

Messages regarding data conflicts

Mode	Conflict Resolution	Conflict Detected?	Sample Message
Preview	Overwrite	Yes	`Conflict: Release 'App1:Release1' will be updated.`
Preview	Skip	Yes	Conflict: Release 'App1:Release1' already exists. will skip.
Preview	Overwrite or Skip	No	`Import: Release 'App1:Release1' will be created.`
Import	Overwrite	Yes	`Updated Script 'Windows Code Rollback' based on importConflict 'overwrite' option.`
Import	Overwrite	Yes	`Deleted Component 'Script 1 (Sample Application:Q3 Release:Tomcat 6.0.20)' based on importConflict 'overwrite' option.`
Import	Skip	Yes	`Lifecycle 'Standard' already exists. Will not import based on importConflict 'skip' option.`
Import	Overwrite or Skip	No	`Created Lifecycle 'Complete'.`

Preview examples

The following simplified examples are intended to illustrate how full and partial import and export operations work. The XML file only contains application definitions. A typical file would also contain information about application groups, releases, lifecycles, environments, components, scripts, tiers, tier groups, and source types.

The preview (-p) operation shows you whether information in an XML file can be successfully imported and whether any conflicts exist.

In this example, the import file contains four applications: MyAppA, MyAppB, MyAppC, and MyAppD. The MyAppA application already exists on this SA core.

Here are the contents of this XML import file (myImportData.xml):

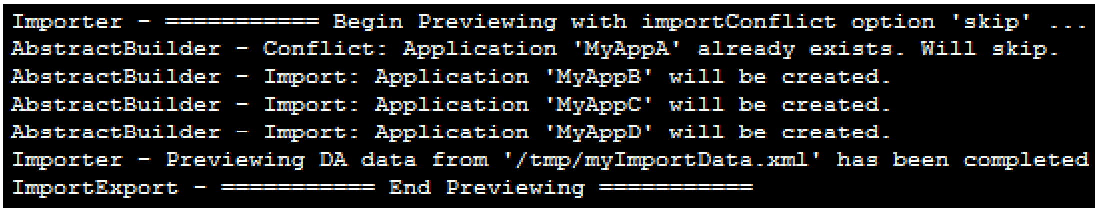
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<DAData xmlns="http://www.hp.com/DA">
       <ModelVersion>1.3</ModelVersion>
       <ReleaseVersion>09.003.001</ReleaseVersion>
       <SCM_VERSION>0</SCM_VERSION>
       <Applications>
               <Application>
                       <Name>MyAppA</Name>
                       <Description>This application exists</Description>
               </Application>
       <Application>
               <Name>MyAppB</Name>
               <Description>This is a new application - no conflict</Description>
       </Application>
       <Application>
               <Name>MyAppC</Name>
               <Description>This is a new application - no conflict</Description>
       </Application>
       <Application>
               <Name>MyAppD</Name>
               <Description>This is a new application - no conflict</Description>
       </Application>
       </Applications>
</DAData>
```

Example 1: Preview of full import

The following command previews the import of all Application Deployment information in the `myImportData.xml` file:

```
/opt/opsware/da/bin/admtool.sh -p --importFile /tmp/myImportData.xml
```

The following messages appear on `stdout` when you perform the preview:



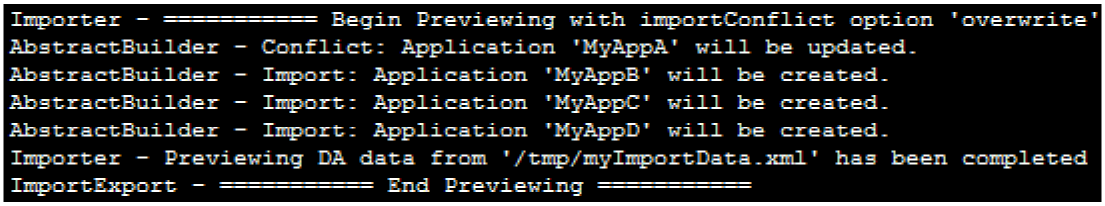
```
Importer - =========== Begin Previewing with importConflict option 'skip' ...
AbstractBuilder - Conflict: Application 'MyAppA' already exists. Will skip.
AbstractBuilder - Import: Application 'MyAppB' will be created.
AbstractBuilder - Import: Application 'MyAppC' will be created.
AbstractBuilder - Import: Application 'MyAppD' will be created.
Importer - Previewing DA data from '/tmp/myImportData.xml' has been completed
ImportExport - =========== End Previewing ===========
```

In this case, MyAppA will not be imported because it already exists on this SA core, and the default conflict resolution strategy (skip) applies.

To use the overwrite conflict resolution strategy, specify this command instead:

```
/opt/opsware/da/bin/admtool.sh -p --importFile /tmp/myImportData.xml --
importConflict overwrite
```

In this case, the following messages appear on `stdout`:

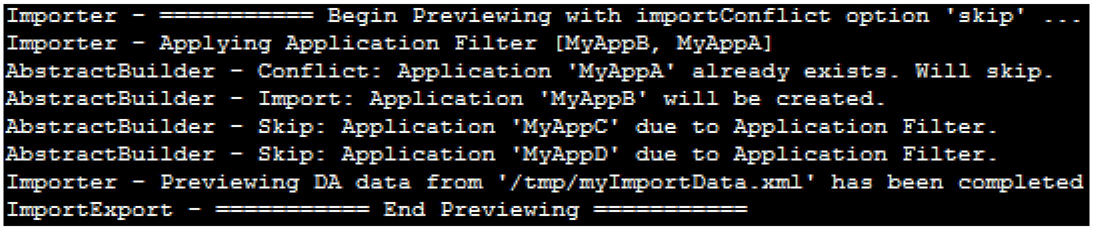


```
Importer - =========== Begin Previewing with importConflict option 'overwrite'
AbstractBuilder - Conflict: Application 'MyAppA' will be updated.
AbstractBuilder - Import: Application 'MyAppB' will be created.
AbstractBuilder - Import: Application 'MyAppC' will be created.
AbstractBuilder - Import: Application 'MyAppD' will be created.
Importer - Previewing DA data from '/tmp/myImportData.xml' has been completed
ImportExport - =========== End Previewing ===========
```

Example 2: Preview of partial import

The following command previews the import of MyAppA and MyAppB:

```
/opt/opsware/da/bin/admtool.sh -p --importAppList "MyAppA;MyAppB" --importFile
/tmp/myImportData.xml
```



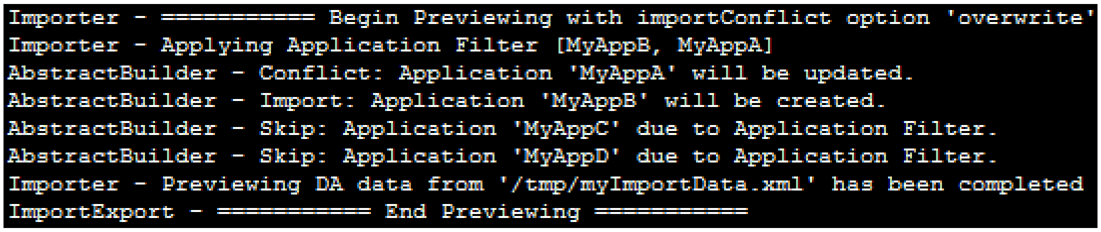
```
Importer - =========== Begin Previewing with importConflict option 'skip' ...
Importer - Applying Application Filter [MyAppB, MyAppA]
AbstractBuilder - Conflict: Application 'MyAppA' already exists. Will skip.
AbstractBuilder - Import: Application 'MyAppB' will be created.
AbstractBuilder - Skip: Application 'MyAppC' due to Application Filter.
AbstractBuilder - Skip: Application 'MyAppD' due to Application Filter.
Importer - Previewing DA data from '/tmp/myImportData.xml' has been completed
ImportExport - =========== End Previewing ===========
```

In this case, MyAppA will not be imported because it already exists on this SA core, and the default (skip) conflict resolution strategy applies. MyAppC and MyAppD will not be imported, because they were not part of the `--importAppList` argument.

To preview this import using the overwrite conflict resolution strategy, use this command:

```
/opt/opsware/da/bin/admtool.sh -p --importAppList "MyAppA;MyAppB" --importFile
/tmp/myImportData.xml --importConflict overwrite
```

In this case, the following messages appear on `stdout`:



```
Importer - =========== Begin Previewing with importConflict option 'overwrite'
Importer - Applying Application Filter [MyAppB, MyAppA]
AbstractBuilder - Conflict: Application 'MyAppA' will be updated.
AbstractBuilder - Import: Application 'MyAppB' will be created.
AbstractBuilder - Skip: Application 'MyAppC' due to Application Filter.
AbstractBuilder - Skip: Application 'MyAppD' due to Application Filter.
Importer - Previewing DA data from '/tmp/myImportData.xml' has been completed
ImportExport - =========== End Previewing ===========
```

Import examples

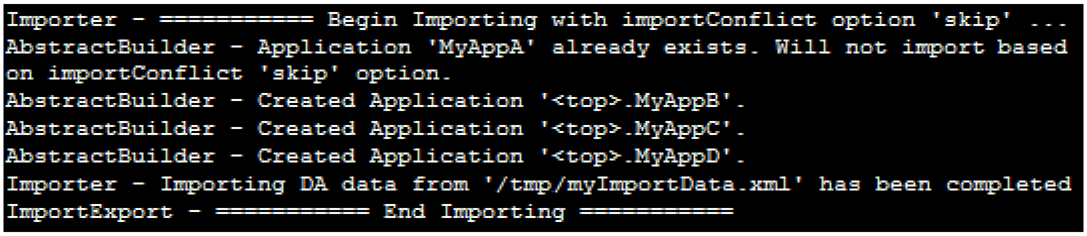
We will continue the previous example and import the contents of the same XML file (`myImportData.xml`).

Example 1: Full import

The following command will attempt to import all the Application Deployment information in the `myImportData.xml` file using the default (skip) conflict resolution strategy:

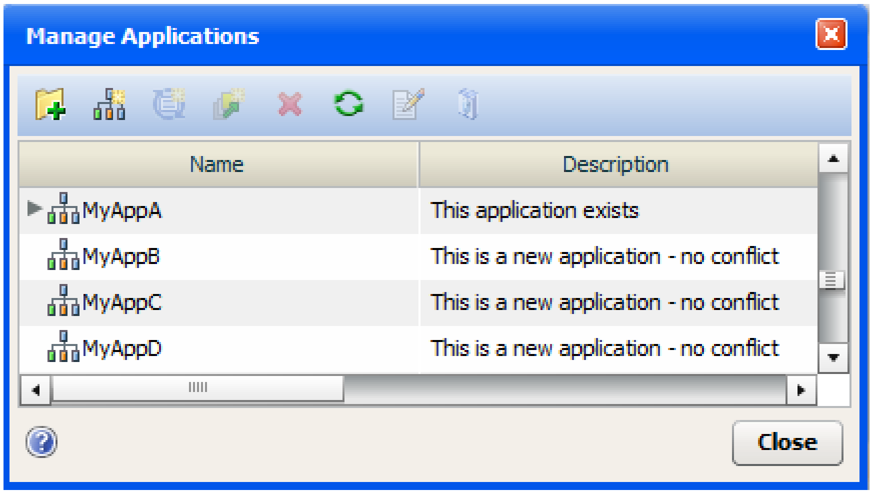
```
/opt/opsware/da/bin/admtool.sh -i --importFile /tmp/myImportData.xml
```

We expect that MyAppA will be skipped, because it already exists, and the other applications (MyAppB, MyAppC, and MyAppD) will be created. The following messages confirm this:



```
Importer - =========== Begin Importing with importConflict option 'skip' ...
AbstractBuilder - Application 'MyAppA' already exists. Will not import based
on importConflict 'skip' option.
AbstractBuilder - Created Application '<top>.MyAppB'.
AbstractBuilder - Created Application '<top>.MyAppC'.
AbstractBuilder - Created Application '<top>.MyAppD'.
Importer - Importing DA data from '/tmp/myImportData.xml' has been completed
ImportExport - =========== End Importing ===========
```

After the import operation, all four applications are listed in the Manage Applications dialog:



Name	Description
MyAppA	This application exists
MyAppB	This is a new application - no conflict
MyAppC	This is a new application - no conflict
MyAppD	This is a new application - no conflict

Note the following:

If you attempt to import data from a file that contains invalid XML, the import will not be performed.

If you import a Software Policy, Package, or Application Configuration component, and the pertinent library items are missing, a message like this one appears:

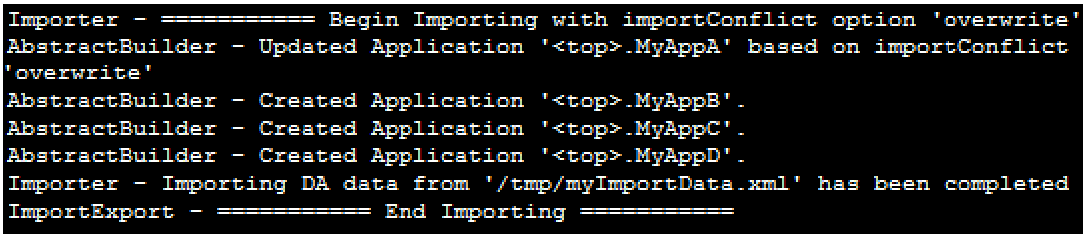
```
Not Found:Package'OPSWpytwist2-40.0.0.7.325.zip(SunOS 5.8)'does not exist
```

If this happens, you will not be able to create a version until you select a different library item—or add the missing item to the library on this core, and then select that item.

To perform this import using the overwrite conflict resolution strategy, use this command:

```
/opt/opsware/da/bin/admtool.sh -i --importFile /tmp/myImportData.xml --
importConflict overwrite
```

In this case, the following messages appear on `stdout`:



```
Importer - =========== Begin Importing with importConflict option 'overwrite'
AbstractBuilder - Updated Application '<top>.MyAppA' based on importConflict
'overwrite'
AbstractBuilder - Created Application '<top>.MyAppB'.
AbstractBuilder - Created Application '<top>.MyAppC'.
AbstractBuilder - Created Application '<top>.MyAppD'.
Importer - Importing DA data from '/tmp/myImportData.xml' has been completed
ImportExport - =========== End Importing ===========
```

Example 2: Partial import

Assuming that only MyAppA exists, the following command imports only MyAppB from the `myImportData.xml` file:

```
/opt/opsware/da/bin/admtool.sh -i --importAppList "MyAppA;MyAppB" --importFile
/tmp/myImportData.xml
```

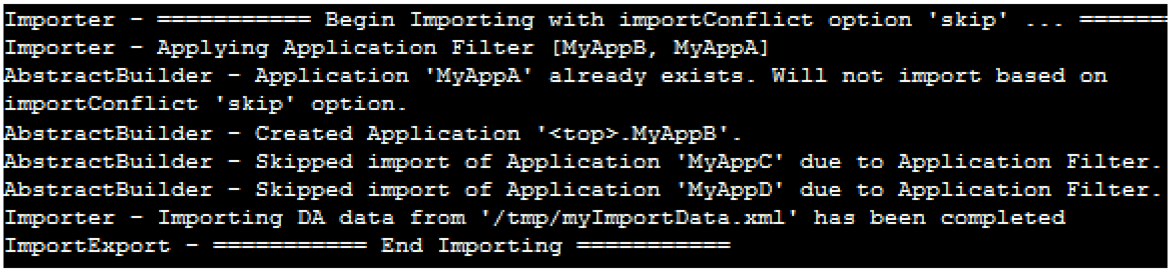
We expect the following results:

MyAppA will be skipped, because it already exists on this SA core.

MyAppB will be created.

MyAppC and MyAppD will not be created, because they are not part of the `--importAppList` argument.

The following messages confirm this result:

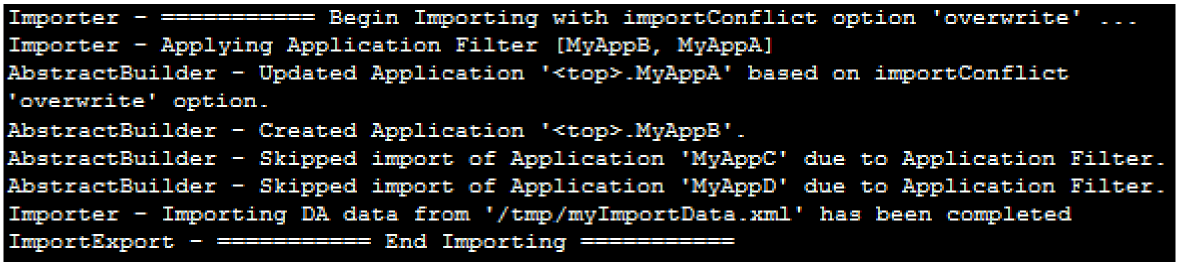


```
Importer - =========== Begin Importing with importConflict option 'skip' ... ======
Importer - Applying Application Filter [MyAppB, MyAppA]
AbstractBuilder - Application 'MyAppA' already exists. Will not import based on
importConflict 'skip' option.
AbstractBuilder - Created Application '<top>.MyAppB'.
AbstractBuilder - Skipped import of Application 'MyAppC' due to Application Filter.
AbstractBuilder - Skipped import of Application 'MyAppD' due to Application Filter.
Importer - Importing DA data from '/tmp/myImportData.xml' has been completed
ImportExport - =========== End Importing ===========
```

To perform this import using the overwrite conflict resolution strategy, use this command:

```
/opt/opsware/da/bin/admtool.sh -i --importAppList "MyAppA;MyAppB" --importFile
/tmp/myImportData.xml --importConflict overwrite
```

In this case, the following messages appear on `stdout`:



```
Importer - =========== Begin Importing with importConflict option 'overwrite' ...
Importer - Applying Application Filter [MyAppB, MyAppA]
AbstractBuilder - Updated Application '<top>.MyAppA' based on importConflict
'overwrite' option.
AbstractBuilder - Created Application '<top>.MyAppB'.
AbstractBuilder - Skipped import of Application 'MyAppC' due to Application Filter.
AbstractBuilder - Skipped import of Application 'MyAppD' due to Application Filter.
Importer - Importing DA data from '/tmp/myImportData.xml' has been completed
ImportExport - =========== End Importing ===========
```

Export examples

There are two types of export operations. A full export writes all the Application Deployment data that exists on this SA core to the specified XML file. A partial export writes only the information that pertains to the specified applications.

For a list of items that can be exported, see "Working" on page 614.

For a list of items that cannot be exported, see "Special considerations" on page 614.

Example 1: Full export

The following command will export all the Application Deployment on this SA core:

```
/opt/opsware/da/bin/admtool.sh -e --exportFile /tmp/myFullExportData.xml
```

Example 2: Partial export

The following command will export only MyAppA and MyAppB:

```
/opt/opsware/da/bin/admtool.sh -e --exportAppList "MyAppA;MyAppB" --exportFile
/tmp/myPartialExportData.xml
```

Assuming that no changes were made to MyAppA or MyAppB since the previous import, the resulting `<Applications>` portion of the `myPartialExportData.xml` file would look like this:

```
<Applications> portion of the myPartialExportData.xml file would look like this:
       <Applications>
               <Application>
                       <Name>MyAppB</Name>
                       <Description>This is a new application - no conflict</Description>
       </Application>
       <Application>
```

```
                        <Name>MyAppA</Name>
                        <Description>This application exists</Description>
                </Application>
</Applications>
```

Send documentation feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on Developer Guide (Server Automation 10.50)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to hpe_sa_docs@hpe.com.

We appreciate your feedback!