Oracle® Virtual Assembly Builder

Developer's Guide

11*g* Release 1 (11.1.1.6)

E26204-02

February 2012

ORACLE®

Oracle Virtual Assembly Builder Developer's Guide, 11*g* Release 1 (11.1.1.6)

E26204-02

Contents

3 API Reference: Administrative Operations

4 API Reference: Deployer Operations

Preface

This book details conceptual, topology and configuration topics about Oracle Virtual Assembly Builder. This Preface includes the following topics:

Audience

The intended audience is developers who create applications leveraging the Web Service API in Oracle Virtual Assembly Builder to create PaaS-enabled applications.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Related Documents

For more information, see the following documents in the documentation set:

- *Oracle Virtual Assembly Builder Installation Guide*
- *Oracle Virtual Assembly Builder User's Guide*
- *Oracle Virtual Assembly Builder Release Notes*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Overview of Oracle Virtual Assembly Builder Deployer

The following sections introduce the Web Service API and its development processes:

1.1 Introduction

The Oracle Virtual Assembly Builder Deployer is a J2EE application that maintains a repository of *assembly archives*, and manages deployment aspects of the software system contained therein. You create assembly archives in Oracle Virtual Assembly Builder Studio.

The Deployer provides operations for registering the assembly archives to virtualized systems such as Oracle VM and for orchestrating the deployment of the software system defined by the assembly archive.

The assembly archives created by Oracle Virtual Assembly Builder Studio contain information about a software system comprised of multiple, related software stacks which work together to form an application. This system is referred to as an assembly. The assembly archive contains metadata about the assembly and virtual machine templates that are used to instantiate an instance of the assembly in a virtualized environment.

The interface to the Oracle Virtual Assembly Builder Deployer is a Web service which provides operations for uploading assembly archives to its repository, registering the assembly archive with the virtualization system and managing deployment instances for the system defined in the assembly archive.

Oracle Virtual Assembly Builder provides additional interfaces as part of the complete product solution. Specifically, Oracle Virtual Assembly Builder Studio and the `abctl` command-line interface integrate with the Web Services.

As described in Section 1.4, "Targets", Oracle Virtual Assembly Builder Deployer can deploy to targets in Oracle VM or Oracle Exalogic. The minor functional differences between these environments are described throughout the documentation.

1.2 Deployment Lifecycle

This section describes the deployment lifecycle including the uploading of assembly archives to the Deployer repository, and the deployment phases of an assembly instance.

1.2.1 Uploading Assembly Archives

Oracle Virtual Assembly Builder Deployer is a Web service which provides operations for uploading assembly archives to its repository.

1.2.1.1 Versioning of Assembly Archives

The Deployer repository allows you to keep multiple versions of an assembly archive in the repository at the same time. When an assembly archive is uploaded, the Deployer will assign a version number to it.

The versioning format increments in whole number units (for example, versions 1, 2, 3, and so on). The latest version is generally considered the default for assembly operations that take a version. If another assembly archive of the same name is uploaded, then it will get the next version.

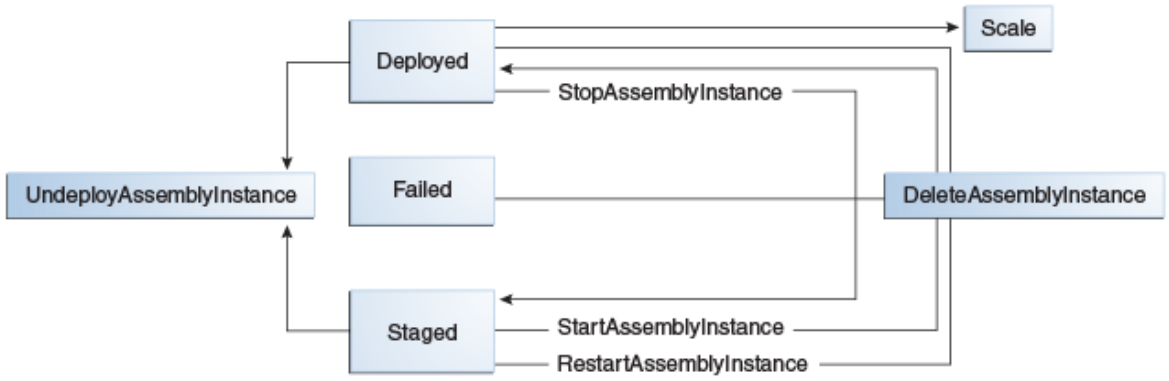
1.2.2 Deployment Phases

An assembly instance is a deployable artifact. You would need to create an assembly instance by selecting an assembly, one of the its deployment plans and the target to which it must be deployed to. `CreateAssemblyInstance` can be used to create the assembly instance.

At deployment time, you choose the assembly instance to be deployed. Deployment of an assembly instance will transition through various phases (Figure 1–1). The phases include: *Staged*, *Deployed*, and *Failed*. Each state allows a subset of operations.

An appliance instance is an instance of an appliance running and/or created in the target virtual environment. When an assembly instance is deployed, you may start and stop the appliance instances, or you may increase or decrease the number of appliance instances associated with that deployed assembly instance. Oracle Virtual Assembly Builder does not monitor the health of the deployed application; it will only inform you of whether or not an assembly is deployed or staged, as well as the success or failure of a deployment-related operation.

Figure 1–1 Deployment Life Cycle



Here is a summary of the assembly instance phases:

- *Deployed* When the assembly instance is deployed and the operation has successfully completed, it reaches the deployed state. The operations that can be performed on a *Deployed* assembly instance are:
  - *StopAssemblyInstance* This operation will shut down all the running appliance instances for the assembly instance. The assembly instance is transitioned to the *Staged* phase after this operation is completed. It leaves the appliance instances in the virtualized environment so that they can be restarted later.
  - *UndeployAssemblyInstance* This operation will stop all the running appliance instances and remove them from the environment. After this operation is completed, the assembly instance will kept in the system so that it can be deployed again.
  - *RestartAssemblyInstance* This operation will restart all the running appliance instances of the assembly instance. The assembly instance will transition to the *Staged* and then transition back to *Deployed*.
  - *RedeployAssemblyInstance* This operation will redeploy the assembly instance. As part of this operation all appliance instances will be stopped and removed from the target environment. New appliance instances will be created and started.
  - *Scale* Scales the scaling group within an assembly instance. Scaling can be performed to scale up or down a scaling group with the assembly instance. The number of appliance instances that can be running for a scaling group must lie between its configured minimum and maximum instance limits. The Deployment continues to remain in the *Deployed* state.
- *Failed* When there is a failure in a deploy or undeploy operation, the assembly instances reaches this phase. A deployment operation may fail for a variety of reasons, such as insufficient resources. The operations that can be performed on a failed deployment are:
  - *DeleteAssemblyInstance* This operation will do the necessary cleanup (such as stopping and removing the appliance instances). After this operation is completed, the assembly instance no longer exists.
- *Staged* The staged phase is reached by stopping an assembly instance. In this phase all the appliance instances have been shut down. The operations that can be performed from this phase are:
  - *StartAssemblyInstance* This operation will start up all the appliance instances that have been shut down. After this operation is completed, the assembly instance is returned to the *Deployed* state.
  - *UndeployAssemblyInstance* This operation will remove all the appliance instances that have been shut down from the virtualized environment. After this operation is completed, the assembly instance will be kept around so that it can be deployed again.

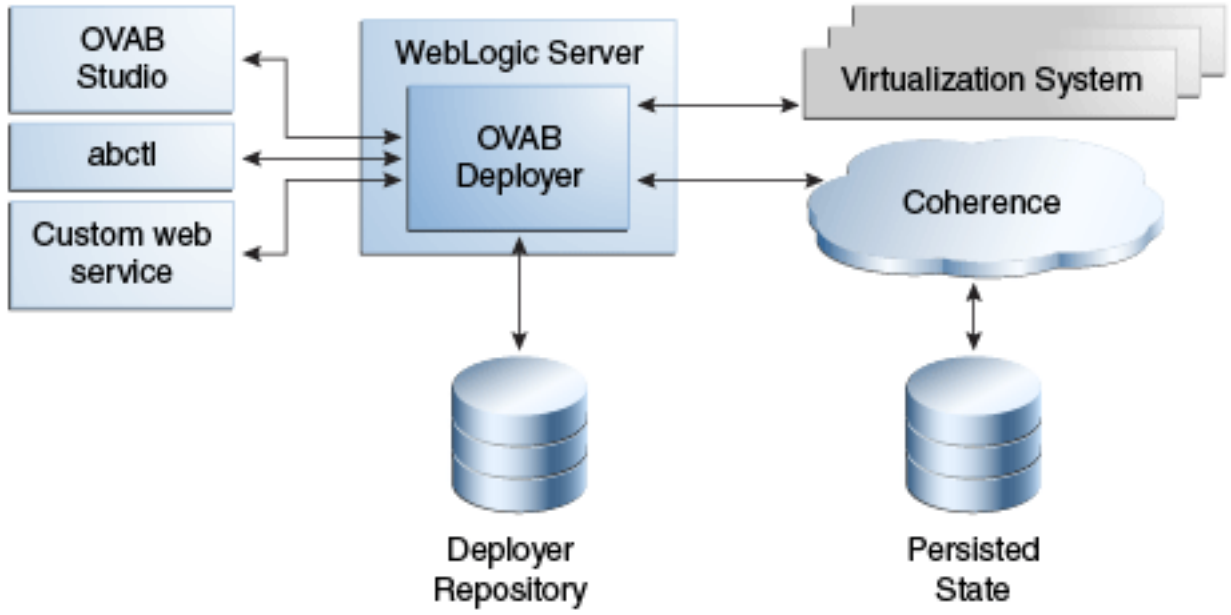
1.3 Deployer Architecture

Oracle Virtual Assembly Builder Deployer is configured as a Web application in a WebLogic Server domain.

Figure 1–2 shows the top-level components of the Oracle Virtual Assembly Builder Deployer. The Deployer Repository contains the assembly archives and any deployment plans uploaded to the Deployer by the Web client. Assembly archives and deployment plans are created in Oracle Virtual Assembly Builder Studio, however, the

repository maintained by the Deployer is separate from the Oracle Virtual Assembly Builder Studio catalog. Plans created in Oracle Virtual Assembly Builder Studio are uploaded to the Deployer through the Web service API. Runtime and configuration state for the Deployer is maintained in a distributed Coherence cache that is persisted so that single instance installations may recover in the case of a failure (that is, the Deployer process exits). The Deployer interacts with one or more virtualization systems and orchestrates the deployment of assembly archives into these systems.

Figure 1–2 Deployer Architecture



1.4 Targets

Different virtualization systems organize their resources in different ways and require different information for referencing and accessing them. In order to provide a common user experience across different systems, Oracle Virtual Assembly Builder Deployer defines the notion of a *target*. Targets are configured using administration interfaces defined later in this document and are used to reference a resource or pool of resources in the virtualized system. The configuration information provided for each target is specific to the virtualization system containing the target.

Oracle Virtual Assembly Builder Deployer provides support for deploying assembly instances to targets on Oracle VM, or Oracle Exalogic.

1.4.1 Oracle Exalogic

Oracle Virtual Assembly Builder Deployer is a key component of Exalogic Control, the management surface area for interacting with Oracle Exalogic. When running in the context of Exalogic Control, Oracle Virtual Assembly Builder Deployer is preconfigured and instantiated via the Exalogic Configuration Utility (ECU) and executes as a virtual machine running within Exalogic Control. All of the functional capabilities of Oracle Virtual Assembly Builder Deployer described here are available when executing in the context of Exalogic Control, and in many cases features and optimizations unique to Exalogic are also present.

See *Exalogic Elastic Cloud Administration and Deployment Guide*.

1.5 Tags

Certain virtualization systems allow tags to be associated with artifacts such as templates and VM instances which may be queried by tools interacting with the system to locate artifacts or to find relationships between artifacts, for example finding the artifacts that are associated with a particular deployment.

In systems that support tagging, the Oracle Virtual Assembly Builder Deployer automatically adds tags to certain artifacts it creates. Table 1–1 provides information on these tags. The values for the tags are always strings, but certain ones may be interpreted as other types. The *Data Type* column indicates these types. For example, if the *Data Type* is Integer, then the value will be a string like "123", which can be interpreted as the number 123. The *Oracle VM Artifact* column indicates what kind of artifact or artifacts a particular tag is to be associated with.

Table 1–1 Oracle Virtual Assembly Builder Deployer Tags

Key Name	Data Type	Oracle VM Artifact	Description	Example
ovab:AssemblyArchiveName	String	Instance, template	The name of the assembly archive used to create the template that is the source of the instance	MySite.ova
ovab:TemplateId	String	Instance	The Oracle VM name of the template that is the source of the instance.	234jlk234jj3lkj43
ovab:TemplateRepositoryName	String	Instance	The name of the repository containing the template that is the source of the instance.	MyRepository
ovab:AppliancePath	String	Instance	The path to the appliance within the Oracle Virtual Assembly Builder assembly. This is used to determine the instance's place in the original assembly structure.	/MyWls/EjbCluster
ovab:ApplianceInstanceIndex	Integer	Instance	The appliance instance index.	2
ovab:ComponentType	String	Instance	The appliance type determined by Oracle Virtual Assembly Builder introspection.	WLS
ovab:DeployerId	String	Instance, assembly archive, template	An identifier for the Deployer instance that created the artifact.	
ovab:AssemblyInstanceId	String	Instance	An identifier for the assembly instance that is unique within the scope of the Deployer instance	Lkj234lk32j4lkj34
ovab:DeploymentPlan	String	Instance	The name of the deployment plan used for the deployment that the instance is a part of.	MyPlan
ovab:UserName	String	Template, assembly archive, instance	The name of the user that initiated the deployment operation that lead to the creation of the artifact.	John
ovab:InitialTargetCount	Integer	Instance	The initial target for the number of instances at deployment time.	5
ovab:ScalabilityMin	Integer	Instance	The minimum number of instances allowed for the appliance.	1
ovab:ScalabilityMax	Integer	Instance	The maximum number of instances allowed for the appliance.	10

1.6 Metadata-Driven Functionality

The details of the deployment orchestration are driven by OVF metadata which is embedded in the assembly archive that represents the application, and information stored in a deployment plan which you provide when creating a deployment.

2

API Web Services

The following sections describes the Oracle Virtual Assembly Builder Web Service API:

- Section 2.1, "Operations"
- Section 2.2, "Administrative Operations"
- Section 2.3, "Usage and Lifecycle"
- Section 2.4, "Admin Web Service"
- Section 2.5, "Deployer Web Service"

2.1 Operations

The Oracle Virtual Assembly Builder Deployer provides logical operations which are used in the Web service and CLI operations. Some operations are asynchronous and the status of these operations can be tracked with a `describeRequests` operation.

The complete API reference for the operations are described in the following chapters:

- Chapter 3, "API Reference: Administrative Operations"
- Chapter 4, "API Reference: Deployer Operations"

2.2 Administrative Operations

Oracle Virtual Assembly Builder Deployer provides logical administrative operations for configuring targets, which reference pools in virtualization environments. All of these operations are synchronous.

Due to the differences in how virtualization environments work and how we interact with them, configuration of these involves two steps.

- Creating targets using the `CreateTarget` operation. This step is only required for Oracle VM, because Oracle Exalogic comes with a single pre-configured target.
- Adding user configuration information for a named user to a target using `AddTargetUser`. For more details, see Section 3.3, "AddTargetUser" in Chapter 3, "API Reference: Administrative Operations".

Operations for adding users are not provided in these administrative operations. It is intended that these operations be performed directly against the LDAP server by security administrators authorized for such duties.

2.3 Usage and Lifecycle

The first operation when using the Oracle Virtual Assembly Builder Deployer is to upload an assembly archive to the Deployer's repository using the `uploadAssemblyArchive` operation.

You can register the uploaded assembly archive with one or more targets using the `registerAssemblyArchive` operation. The details of registration are specific to the target type. For Oracle VM, registration includes uploading the assembly archive to the Deployer and registering the templates with a target within the Oracle VM system.

After an assembly archive has been registered with a target, you can create one or more assembly instances for that registration using the `createAssemblyInstance` operation. An assembly instance is a context used to manage the lifecycle of VM instances for the assembly defined by the assembly archive. When creating an assembly instance, a deployment plan must be provided which defines the environment-specific details of the assembly managed through the assembly instance.

Once an assembly instance has been created for an assembly archive, you can initiate a deployment operation. During deployment, the initial instances for the assembly defined by the assembly archive are created and started. As this is done, the Deployer interacts with reconfiguration logic embedded in the initialization scripts of the instances which configures networks and mounted disk volumes and also configures the application stack for the environment. These interactions configure connections between instances that are created when the application stack starts up.

Once an assembly instance has been deployed, a number of other lifecycle operations can be applied to the running system such as `scale` which increases or decreases the number of running instances for an appliance within the assembly. There are also operations to stop and start all instances of the deployment, which are used for bringing the application offline and back online.

2.4 Admin Web Service

Operations against the Web service are made by posting an HTTP request to the Deployer's context path. The request includes a request parameter that defines the action followed by zero or more request parameters that define arguments for the operation. See Section 2.5.2, "Query String Pattern" below for more details.

2.4.1 Context Path

The context path for the admin Web service is `/ovab/admin`.

2.4.2 Query String Pattern

Action= *action*&param1=*value*&param2=*value*...

For an array of values, use the same parameter name multiple times in the query string. For example:

Action= *action*&param1=*value*&param2=*value*&param2=*value*...

2.4.3 Actions

Chapter 3, "API Reference: Administrative Operations"describes all of the operations exposed by the admin Web service. Details of the types included in the "Response Content" column may be found in the schema in Appendix A, "Web Service Schema".

2.4.4 Error Handling

Table 2–1 shows how error handling is performed for the Web service.

Table 2–1 Web Service Error Handling

Responsible for Error	HTTP Status Code	Response Content XML Element
User	4xx	ErrorResult (type=USER)
Service	5xx	ErrorResult (type=SERVICE)

Note: In the case of an error, there may not be any content (`ErrorResult`) in the response. If the HTTP status code is generated by the servlet container, then there will not be any content in response. If the HTTP response is generated by the Oracle Virtual Assembly Builder Deployer, then most often there will be content in response, except when an unexpected service error occurs.

2.4.5 XML Schema

You can find the schema for the Web service in Appendix A, "Web Service Schema". This schema includes definitions for the response content items included in the table of actions, above.

2.5 Deployer Web Service

Operations against the Web service are made by posting an HTTP request to the Deployer's context path. The request includes a request parameter that defines the action followed by zero or more request parameters that define arguments for the operation. See Section 2.5.2, "Query String Pattern" below for more details.

As described in Section 2.1, "Operations", some operations may initiate an asynchronous action. The operations exposed by the Web service are the same as is described for the logical operations described earlier.

2.5.1 Context Path

The context path for the Deployer Web service is `/ovab/deployer`.

2.5.2 Query String Pattern

Action= *action*&param1=*value*&param2=*value*...

For an array of values, use the same parameter name multiple times in the query string. For example:

Action= *action*&param1=*value*&param2=*value*&param2=*value*...

2.5.3 Actions

Chapter 4, "API Reference: Deployer Operations" describes all of the operations exposed by the Deployer Web service. Details of the types included in the "Response Content" column may be found in the schema in Appendix A, "Web Service Schema".

2.5.4 Error Handling

Table 2–2 shows how error handling is performed for the Deployer Web service.

Table 2–2 Web Service Error Handling

Responsible for Error	HTTP Status Code	Response Content XML Element
User	4xx	ErrorResult (type=USER)
Service	5xx	ErrorResult (type=SERVICE)

Note: In the case of an error, there may not be any content (`ErrorResult`) in the response. If the HTTP status code is generated by the servlet container, then there will not be any content in response. If the HTTP response is generated by the Oracle Virtual Assembly Builder Deployer, then most often there will be content in response, except when an unexpected service error occurs.

2.5.5 XML Schema

You can find the schema for the Web service in Appendix A, "Web Service Schema". This schema includes definitions for the response content items included in the table of actions, above.

3

API Reference: Administrative Operations

The following sections describes administrative operations in the Oracle Virtual Assembly Builder Deployer Web Service API:

3.1 Parameters in HTTP Query String

The following designations apply to parameters in the "Other Parameters in HTTP Query String" sections of each operation:

- 1..1 : Parameter is required, and can have exactly one occurrence.
- 1..* : Parameter is required, and can have multiple occurrences.
- 0..1 : Parameter is optional, and can have exactly one occurrence, if provided.
- 0..* : Parameter is optional, and can have multiple occurrences, if provided.

3.2 Response Content

The response types for the operations correspond to definitions in the Web service schema. See Appendix A, "Web Service Schema".

3.3 AddTargetUser

This operation adds user configuration information for the named user to the target. The purpose of `AddTargetUser`, in addition to indicating that a user will be using the target, is to supply user specific information to be used when interacting with the target, if required.

For the Deployer which is used with Oracle VM, `AddTargetUser` is called by the Cloud Administrator to grant access to the target. In this case, no other properties are provided because the Cloud Administrator provides credentials for the virtualization environment when the target is created. Application Administrators are not allowed to call this operation for the generic Deployer because the credentials must be protected.

For the Deployer in Oracle Exalogic, `AddTargetUser` is called by the Application Administrator who will be using the target. In this case, the user provides his or her own credentials for accessing the target.

Other Parameters in HTTP Query String

target 1..1

user 1..1

Other parameters are used as properties for the user

HTTP Method

GET

Request Content

N/A

Response Content

text/xml Element: AddTargetUserResult

Required Role and Authorization

In the Oracle VM Deployer, only the Cloud Administrator is allowed to call `AddTargetUser`. In the Oracle Exalogic Deployer, any Application Administrator can call `AddTargetUser`.

The reason for this difference is that in the generic case, the credentials are provided in the target configuration and must be protected, but in the Oracle Exalogic Deployer, the user supplies their own credential.

A Cloud Administrator who calls this operation can specify a username other than their own.

3.4 CreateTarget

This operation creates a target for deployment operations. The call includes configuration information that defines the connection information, but depending on the type of target, credential information may or may not be included. Specifically, for the Oracle VM 3.0 target, credentials are supplied, but for the Oracle Exalogic target, credentials are not supplied here and you must supply them later in the `AddTargetUser` operation.

Other Parameters in HTTP Query String

name 1..1

type 1..1

default 0..1 (default value is false)

Other parameters are used as properties for the target

HTTP Method
GET

Request Content
N/A

Response Content
text/xml Element: CreateTargetResult

Required Role and Authorization
This operation can only be called by the Cloud Admin.

3.5 DeleteTarget

This operation deletes the deployment target and all configuration information. If this target was a default for a user or all users, then that default is unset.

Other Parameters in HTTP Query String
name 1..1

HTTP Method
GET

Request Content
N/A

Response Content
text/xml Element: DeleteTargetResult

Required Role and Authorization
This operation can only be called by the Cloud Admin.

3.6 DescribeTargetConfigurations

This operation describes configuration information for a target.

Other Parameters in HTTP Query String
target 1..1

HTTP Method
GET

Request Content
N/A

Response Content
text/xml Element: DescribeTargetConfigurationsResult

Required Role and Authorization
This operation can only be called by the Cloud Admin.

3.7 DescribeTargetNames

This operation lists created targets for deployment operations for the given type.

Other Parameters in HTTP Query String
type 1..1

HTTP Method
GET

Request Content
N/A

Response Content
text/xml Element: ListTargetsResult

Required Role and Authorization
This operation can be called by the Application Admin or Cloud Admin.

3.8 DescribeTargetUsers

This operation lists users for the given target.

Other Parameters in HTTP Query String
target 1..1

HTTP Method
GET

Request Content
N/A

Response Content
text/xml Element: DescribeTargetUsersResult

Required Role and Authorization
This operation can be called by the Application Admin or Cloud Admin.

3.9 DescribeUserTargets

This operation lists targets for the given user.

Other Parameters in HTTP Query String
user 1..1

HTTP Method
GET

Request Content
N/A

Response Content
text/xml Element: DescribeUserTargetsResult

Required Role and Authorization
This operation can be called by the Application Admin or Cloud Admin.

3.10 GetDefaultTarget

This operation gets the default target of that type for all users.

Other Parameters in HTTP Query String
type 1..1

HTTP Method
GET

Request Content
N/A

Response Content
text/xml Element: GetDefaultTargetResult

Required Role and Authorization
This operation can be called by the Application Admin or Cloud Admin.

3.11 GetTargetType

This operation gets the target type (`exalogic`, or `ovm`) for the given target name.

Other Parameters in HTTP Query String
target 1..1

HTTP Method
GET

Request Content
N/A

Response Content
text/xml Element: GetTargetTypeResult

Required Role and Authorization
This operation can be called by the Application Admin or Cloud Admin.

3.12 RemoveTargetUsers

This operation removes a user from a target.

Other Parameters in HTTP Query String

target 1..1

user 1..*

HTTP Method

GET

Request Content

N/A

Response Content

text/xml Element: RemoveTargetUsersResult

Required Role and Authorization

This operation can be called by the Application Admin or Cloud Admin.

A caller holding the Cloud Admin role can call the method and specify a username other than their own. A non-admin user can only specify their own username.

3.13 SetDefaultTarget

This operation sets the target type of the given name as the default target for all targets of that type for all users. The specified target is used if the user calls an Oracle Virtual Assembly Builder operation and has neither specified a target nor has previously called `setUserDefaultTarget` for the target type.

Other Parameters in HTTP Query String

name 1..1

HTTP Method

GET

Request Content

N/A

Response Content

text/xml Element: SetDefaultTargetResult

Required Role and Authorization

This operation can only be called by the Cloud Admin.

4

API Reference: Deployer Operations

The following sections describes Deployer operations in the Oracle Virtual Assembly Builder Deployer Web Service API which are used to manage assembly archives and deployment aspects of the software system contained therein:

4.1 Parameters in HTTP Query String

The following designations apply to parameters in the "Other Parameters in HTTP Query String" sections of each operation:

- 1..1 : Parameter is required, and can have exactly one occurrence.
- 1..* : Parameter is required, and can have multiple occurrences.
- 0..1 : Parameter is optional, and can have exactly one occurrence, if provided.
- 0..* : Parameter is optional, and can have multiple occurrences, if provided.

4.2 AddAssemblyUsers

This operation grants access to an assembly to other users. This operation can only be called by the assembly owner or Cloud Admin. The owner is defined to be the user who first uploaded the assembly to the Deployer.

A user who is granted access may register the assembly with a target and create an assembly instance.

Other Parameters in HTTP Query String

user 1..*

assembly.name 1..1

HTTP Method

GET

Request Content

N/A

Behavior

Synchronous.

Response Content

text/xml Element: AddAssemblyUsersResult

4.3 CreateAssemblyInstance

This operation creates an assembly instance for an assembly.

Other Parameters in HTTP Query String

assembly.name 1..1

assembly.version 1..1

target 1..1

HTTP Method

POST

Request Content

application/octetstream

Plan file

Behavior

Synchronous.

Response Content

text/xml Element: CreateAssemblyInstanceResult

4.4 CreateTags

This operation creates one or more tags for a resource.

Other Parameters in HTTP Query String

tag 1..*

resource.id 1..1

HTTP Method

GET

Request Content

N/A

Behavior

Synchronous.

Response Content

text/xml Element: CreateTagsResult

4.5 DeleteAssemblyArchive

This operation deletes an assembly archive from the Deployer. This operation may only be performed if there are no registrations for the assembly archive.

Other Parameters in HTTP Query String

assembly.name 1..1

assembly.version 1..1

HTTP Method
GET

Request Content
N/A

Behavior
Synchronous.

Response Content
text/xml Element: DeleteAssemblyArchiveResult

4.6 DeleteAssemblyInstance

This operation deletes an assembly instance. This operation can only be executed when the assembly is in an undeployed state.

Other Parameters in HTTP Query String
assembly.instance.id

1..1

HTTP Method
GET

Request Content
N/A

Behavior
Synchronous.

Response Content
text/xml Element: DeleteAssemblyInstanceResult

4.7 DeleteRequests

This operation deletes one or more previously completed requests.

Other Parameters in HTTP Query String
request.id 1..1

HTTP Method
GET

Request Content
N/A

Behavior
Synchronous.

Response Content
text/xml Element: DeleteRequestsResult

4.8 DeleteTags

This operation deletes one or more tags for a resource.

Other Parameters in HTTP Query String
tag 1..*

resource.id 1..1

HTTP Method
GET

Request Content
N/A

Behavior
Synchronous.

Response Content
text/xml Element: DeleteTagsResult

4.9 DeployAssemblyInstance

This operation deploys an assembly instance. This operation orchestrates the creation of the initial instances for the deployment defined in the OVF and deployment plan.

Other Parameters in HTTP Query String
assembly.instance.id 1..1

HTTP Method
GET

Request Content
N/A

Behavior
Asynchronous.

Response Content
text/xml Element: DeployAssemblyInstanceResult

4.10 DescribeAssemblyArchives

This operation describes one or more assemblies in the Deployer.

Other Parameters in HTTP Query String
assembly.name 1..*

HTTP Method
GET

Request Content
N/A

Behavior
Synchronous.

Response Content
text/xml Element: DescribeAssemblyArchivesResult

4.11 DescribeApplianceInstances

This operation describes one or more deployed instances of an assembly.

Other Parameters in HTTP Query String
assembly.instance.id 0..1

appliance.id 0..1

instance.id 0..1

HTTP Method
GET

Request Content
N/A

Behavior
Synchronous.

Response Content
text/xml Element: DescribeApplianceInstancesResult

4.12 DescribeAssemblyInstances

This operation describes one or more assembly instances.

Other Parameters in HTTP Query String
assembly.instance.id 1..*

HTTP Method
GET

Request Content
N/A

Behavior
Synchronous.

Response Content
text/xml Element: DescribeAssemblyInstancesResult

4.13 DescribeAssemblyUsers

This operation describes one or more users of an assembly.

Other Parameters in HTTP Query String
assembly.name 1..1

HTTP Method
GET

Request Content
N/A

Behavior
Synchronous.

Response Content
text/xml Element: DescribeAssemblyUsersResult

4.14 DescribeDeployer

This operation provides version information for the Deployer.

Other Parameters in HTTP Query String
N/A

HTTP Method
GET

Request Content
N/A

Behavior
Synchronous.

Response Content
text/xml Element: RedeployAssemblyInstanceResult

4.15 DescribeRegistrations

This operation describes one or more assembly registrations.

Other Parameters in HTTP Query String
assembly.name 1..1

assembly.version 1..1

HTTP Method
GET

Request Content
N/A

Behavior
Synchronous.

Response Content
text/xml Element: DescribeRegistrationsResult

4.16 DescribeRequests

This operation describes one or more previously issued requests.

Other Parameters in HTTP Query String
request.id 1..*

HTTP Method
GET

Request Content
N/A

Behavior
Synchronous.

Response Content
text/xml Element: DescribeRequestsResult

4.17 DescribeScalingGroups

This operation describes one or more scaling groups.

Other Parameters in HTTP Query String
assembly.instance.id 0..*

scalinggroup.id 0..*

HTTP Method
GET

Request Content
N/A

Behavior
Synchronous.

Response Content
text/xml Element: DescribeScalingGroupsResult

4.18 DescribeTags

This operation describes one or more tags for a resource.

Other Parameters in HTTP Query String
tag 1..*

resource.id 1..1

HTTP Method
GET

Request Content
N/A

Behavior
Synchronous.

Response Content
text/xml Element: DescribeTagsResult

4.19 DescribeTargets

This operation describes one or more assembly instance targets.

Other Parameters in HTTP Query String
target 1..*

HTTP Method
GET

Request Content
N/A

Behavior
Synchronous.

Response Content
text/xml Element: DescribeTargetsResult

4.20 DescribeVnets

This operation describes one or more networks.

Other Parameters in HTTP Query String
target 1..1

id 0..*

HTTP Method
GET

Request Content
N/A

Behavior
Synchronous.

Response Content
text/xml Element: DescribeVnetsResult

4.21 DownloadAssemblyArchive

This operation downloads an assembly from the Deployer.

Other Parameters in HTTP Query String
assembly.name 1..1

assembly.version 1..1

HTTP Method
GET

Request Content
N/A

Behavior
Synchronous.

Response Content
application/octet-stream

assembly archive (.ova file)

4.22 DownloadAssemblyMetadata

This operation downloads assembly metadata descriptor from the Deployer.

Other Parameters in HTTP Query String
assembly.name 1..1

assembly.version 1..1

HTTP Method
GET

Request Content
N/A

Behavior
Synchronous.

Response Content
application/octet-stream

assembly metadata (.ovf file)

4.23 DownloadDeploymentPlan

This operation downloads a deployment plan from an existing deployment.

Other Parameters in HTTP Query String
assembly.instance.id 1..1

HTTP Method
GET

Request Content
N/A

Behavior
Synchronous.

Response Content
application/octet-stream

Plan file

4.24 RedeployAssemblyInstance

This operation redeploys an assembly instance.

Other Parameters in HTTP Query String
assembly.instance.id 1..1

HTTP Method
GET

Request Content
N/A

Behavior
Asynchronous.

Response Content
text/xml Element: RedeployAssemblyInstanceResult

4.25 RegisterAssemblyArchive

This operation registers the archive with the backend virtualization system (Oracle VM or Oracle Exalogic).

Other Parameters in HTTP Query String
assembly.name 1..1

assembly.version 1..1

assembly.desc 1..1

HTTP Method
GET

Request Content
N/A

Behavior
Synchronous.

Response Content
text/xml Element: RegisterAssemblyArchiveResult

4.26 RemoveAssemblyUsers

This operation removes users from an assembly.

Other Parameters in HTTP Query String
user 1..*

assembly.name 1..1

HTTP Method
GET

Request Content
N/A

Behavior
Asynchronous.

Response Content
text/xml Element: RemoveAssemblyUsersResult

4.27 RestartAssemblyInstance

This operation restarts an assembly instance.

Other Parameters in HTTP Query String
assembly.instance.id 1..1

HTTP Method
GET

Request Content
N/A

Behavior
Asynchronous.

Response Content
text/xml Element: RestartAssemblyInstanceResult

4.28 ScaleAppliance

This operation scales an appliance. This operation changes the number of instances desired for an appliance and will lead to instances being created and started or stopped and destroyed depending on the new number.

Other Parameters in HTTP Query String
scalinggroup.id 1..1

target 1..1

HTTP Method
GET

Request Content
N/A

Behavior
Asynchronous.

Response Content
text/xml Element: ScaleApplianceResult

4.29 StartAssemblyInstance

This operation starts an assembly instance.

Other Parameters in HTTP Query String
deployment.id 1..1

HTTP Method
GET

Request Content
N/A

Behavior
Asynchronous.

Response Content
text/xml Element: StartAssemblyInstanceResult

4.30 StopAssemblyInstance

This operation stops an assembly instance.

Other Parameters in HTTP Query String
assembly.instance.id 1..1

force 0..1 (default is false)

HTTP Method
GET

Request Content
N/A

Behavior
Asynchronous.

Response Content
text/xml Element: StopAssemblyInstanceResult

4.31 UndeployAssemblyInstance

This operation undeploys the assembly instance.

Other Parameters in HTTP Query String
assembly.instance.id 1..1

HTTP Method
GET

Request Content
N/A

Behavior
Synchronous.

Response Content
text/xml Element: UndeployAssemblyInstanceResult

4.32 UnregisterAssemblyArchive

This operation removes the registration artifacts from the backend virtualization system (target). This operation can only be called if there are no assembly instances for this assembly archive in the target.

Other Parameters in HTTP Query String
assembly.version 1..1

assembly.name 1..1

target 1..1

HTTP Method
GET

Request Content
N/A

Behavior
Synchronous.

Response Content
text/xml Element: UnregisterAssemblyArchiveResult

4.33 UpdateAssemblyArchive

This operation allows you to modify the description information for the assembly archive.

> **Note:** To upload a new version, you would call `UploadAssemblyArchive` again.

Other Parameters in HTTP Query String
assembly.name 1..1

assembly.version 1..1

assembly.desc 1..1

HTTP Method
GET

Request Content
N/A

Behavior
Synchronous.

Response Content
text/xml Element: UpdateAssemblyArchiveResult

4.34 UploadAssemblyArchive

This operation is used to upload an assembly operation from a client to the Deployer. The Deployer will keep multiple versions of an assembly archive so you may call this operation more than once for the same assembly.

Other Parameters in HTTP Query String
assembly.name 1..1

HTTP Method
POST

Request Content
application/octetstream

Assembly archive (.ova file)

Behavior

Synchronous.

Response Content

text/xml Element: UploadAssemblyArchiveResult

5 Sample Application

The following sections describes a sample application containing the Web Service APIs:

- Section 5.1, "Sample Application"

5.1 Sample Application

The following sample application shows how to perform each of the steps to deploy an assembly archive for a generic Oracle VM 3.0 platform.

```
import java.io.*;
import java.net.HttpURLConnection;
import java.net.URL;
import java.security.KeyStore;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Date;
import java.util.Hashtable;
import java.util.Map;
import java.util.HashMap;
import java.util.LinkedList;
import javax.net.SocketFactory;
import javax.net.ssl.*;
import org.xml.sax.Attributes;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.DefaultHandler;
import org.xml.sax.helpers.XMLReaderFactory;

public class OvabSamples
{

  private static final String OVM_URL = "https://myovm:7001";
  private static final String OVM_POOL_NAME = "mypool";
  private static final String OVM_ADMIN_NAME = "admin";
  private static final String OVM_ADMIN_PWD = "your ovm admin password";
  private static final String TARGET_NAME = "ovm-target";

  private static final String KEYSTORE_FILE = "/ab_home/ab_
instance/config/deployer/DeployerTrust.jks";
  private static final String TRUSTSTORE_FILE = "/ab_home/ab_
instance/config/deployer/DeployerTrust.jks";
  private static final String TRUSTSTORE_PASSWORD = "your truststore password";
```

```
  private static final String ASSEMBLY_NAME = "MyAssembly";

  private static final String OVA_FILE="/archives/BaseWLS.ova";
  private static final String PLAN_FILE="/plans/BaseWLSQAPlan.xml";

  private static final String DEPLOYER_URL = "https://localhost:7002";
  private static final String CLOUD_ADMIN_USERNAME="cloudAdmin";
  private static final String CLOUD_ADMIN_PASSWORD="cloud admin password";
  private static final String APPLICATION_ADMIN_USERNAME="applicationAdmin";
  private static final String APPLICATION_ADMIN_PASSWORD="application
administrator";

  /*
   * This example shows how to deploy an OVA end-to-end on the standard OVAB
Deployer.
   */
  public static void main(String[] args) throws Exception {

    createOvmTarget();
    addOvmTargetUser();
    uploadAssemblyArchive();

    int version = registerAssemblyArchive();
    if (version >= 0) {
      String assemblyInstanceId = createAssemblyInstance(version);
      if (deployAssemblyInstance(assemblyInstanceId)) {
        message("Creation of Assembly Instance Succeeded");
      } else {
        message("Creation of Assembly Instance Failed");
      }
    } else {
      message("Registration Failed");
    }

  }

  /*
   * When using the generic Deployer, you need to define a target.
   * The target includes configuration for the backend system. In
   * this example, the backend system is Oracle VM 3.
   *
   * This operation is not used for the Oracle Exalogic Deployer as the
   * one target for Exalogic systems is pre-defined.
   */
  public static void createOvmTarget() throws Exception {

    message("CREATE Oracle VM target "+TARGET_NAME);

    // Set up connection
    HttpURLConnection conn = getConnection(DEPLOYER_URL+"/ovab/admin");
    conn.setDoInput(true);
    conn.setDoOutput(true);
    conn.setRequestMethod("POST");
    String params =
      "Action=CreateTarget" +
      "&target=" + TARGET_NAME +
      "&type=ovm" +
      "&default=false" +
      "&ovm.poolName=" + OVM_POOL_NAME +
      "&ovm.vmOperationTimeout=600000" +
```

```
      "&ovm.vmmversion=3.0" +
      "&ovm.user=" + OVM_ADMIN_NAME +
      "&ovm.url=" + OVM_URL +
      "&ovm.pwd=" + OVM_ADMIN_PWD;
    conn.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
    conn.setRequestProperty("Content-Length", "" + params.length());

    // Use Cloud Admin Credentials
    sun.misc.BASE64Encoder enc = new sun.misc.BASE64Encoder();
    String encodedCredentials =
      enc.encode(new String(CLOUD_ADMIN_USERNAME + ":" + CLOUD_ADMIN_
PASSWORD).getBytes());
    conn.setRequestProperty("Authorization", "Basic " + encodedCredentials);

    // Make the request
    try {
      conn.connect();
      // send the parameters
      OutputStream os = conn.getOutputStream();
      os.write(params.getBytes("UTF-8"));
      os.close();

      // Read the response
      Map<String, String> info = handleSuccessResponse(conn);
      for (Map.Entry<String, String> entry : info.entrySet()) {
        message("    "+entry.getKey()+"="+entry.getValue());
      }
      message("Create result: "+info);
      message("SUCCESS");

    } catch (Exception e) {
      Map<String, String> info = handleException(conn, e);
      for (Map.Entry<String, String> entry : info.entrySet()) {
        message("    "+entry.getKey()+"="+entry.getValue());
      }
    }

  }

  /*
   * For Oracle VM 3, users must be authorized by the administrator to use a
target
   *
   * For Oracle Exalogic, the user calls this operation to provide  his or her own
credentials
   */
  public static void addOvmTargetUser() throws Exception {

    message("ADD Oracle VM target user "+TARGET_NAME);

    // Set up connection
    HttpURLConnection conn = getConnection(DEPLOYER_URL+"/ovab/admin");
    conn.setDoInput(true);
    conn.setDoOutput(true);
    conn.setRequestMethod("POST");
    String params =
      "Action=AddTargetUser"+
      "&user=" + APPLICATION_ADMIN_USERNAME +
      "&target=" + TARGET_NAME;
    conn.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
```

```
    conn.setRequestProperty("Content-Length", "" + params.length());

    // Use Admin credentials
    sun.misc.BASE64Encoder enc = new sun.misc.BASE64Encoder();
    String encodedCredentials =
      enc.encode(new String(CLOUD_ADMIN_USERNAME + ":" + CLOUD_ADMIN_
PASSWORD).getBytes());
    conn.setRequestProperty("Authorization", "Basic " + encodedCredentials);

    // Make the request
    try {
      conn.connect();
      // send the parameters
      OutputStream os = conn.getOutputStream();
      os.write(params.getBytes("UTF-8"));
      os.close();
      // Read the response
      Map<String, String> info = handleSuccessResponse(conn);
      for (Map.Entry<String, String> entry : info.entrySet()) {
        message("    "+entry.getKey()+"="+entry.getValue());
      }
      message("SUCCESS");

    } catch (Exception e) {
      Map<String, String> info = handleException(conn, e);
      for (Map.Entry<String, String> entry : info.entrySet()) {
        message("    "+entry.getKey()+"="+entry.getValue());
      }
    }

  }


  /*
   * This is an example of how to upload an OVA from the client to the Deployer
repository.
   */
  public static void uploadAssemblyArchive() throws Exception {
    message("UPLOAD OVA");
    // Set up connection
    HttpURLConnection conn =
      getConnection(DEPLOYER_URL+
                    "/ovab/deployer"+
                    "?Action=UploadAssemblyArchive"+
                    "&assembly.name="+ASSEMBLY_NAME+
                    "&assembly.desc="+"my sample assembly");

    conn.setDoInput(true);
    conn.setDoOutput(true);
    conn.setRequestMethod("POST");
    conn.setRequestProperty("Content-Type", "application/octet-stream");
    conn.setChunkedStreamingMode(0);

    sun.misc.BASE64Encoder enc = new sun.misc.BASE64Encoder();
    String encodedCredentials =
      enc.encode(new String(APPLICATION_ADMIN_USERNAME + ":" + APPLICATION_ADMIN_
PASSWORD).getBytes());
    conn.setRequestProperty("Authorization", "Basic " + encodedCredentials);

    // Make the request
```

```
  try {
    conn.connect();

    int bytesRead, bufferSize;
    int maxBufferSize = 1 * 1024 * 1024;
    byte[] buffer;
    File f = new File(OVA_FILE);
    FileInputStream fis = new FileInputStream(f);
    bufferSize = Math.min(fis.available(), maxBufferSize);
    buffer = new byte[bufferSize];
    bytesRead = fis.read(buffer, 0, bufferSize);

    OutputStream os = conn.getOutputStream();
    while (bytesRead > 0) {
      os.write(buffer, 0, bufferSize);
      bufferSize = Math.min(fis.available(), maxBufferSize);
      bytesRead = fis.read(buffer, 0, bufferSize);
    }
    fis.close();
    os.flush();
    os.close();

    // Read the response
    Map<String, String> info = handleSuccessResponse(conn);
    for (Map.Entry<String, String> entry : info.entrySet()) {
      message("    "+entry.getKey()+"="+entry.getValue());
    }
    message("SUCCESS");

  } catch (Exception e) {
    Map<String, String> info = handleException(conn, e);
    for (Map.Entry<String, String> entry : info.entrySet()) {
      message("    "+entry.getKey()+"="+entry.getValue());
    }
  }


}

/*
 * This is an example of how to register an assembly archive from the Deployer
 * repository to the backend system
 *
 * Registration is an asynchronous task, so this method calls
 * another method to wait for the asynchronous request to complete
 * before returning.
 */
public static int registerAssemblyArchive() throws Exception {

  message("REGISTER Assembly Archive");

  // Set up connection
  String params =
    "Action=RegisterAssemblyArchive"+
    "&assembly.name="+ASSEMBLY_NAME+
    "&target="+TARGET_NAME;
  HttpURLConnection conn = getConnection(DEPLOYER_URL+"/ovab/deployer?"+params);
  conn.setDoInput(true);
  conn.setRequestMethod("GET");
  conn.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
```

```
    conn.setRequestProperty("Content-Length", "" + params.length());

    // Use Application Admin Credentials
    sun.misc.BASE64Encoder enc = new sun.misc.BASE64Encoder();
    String encodedCredentials =
      enc.encode(new String(APPLICATION_ADMIN_USERNAME + ":" + APPLICATION_ADMIN_
PASSWORD).getBytes());
    conn.setRequestProperty("Authorization", "Basic " + encodedCredentials);

    // Make the request
    try {
      conn.connect();

      // Read the response
      message("    Async Registration Request Initiated");
      Map<String, String> info = handleSuccessResponse(conn);
      String requestId =  info.get("RegisterAssemblyArchiveResult.requestId");

      // Wait for the request to complete
      if (waitForRequest(requestId)) {
        // return the version registered. This is used later for creating the
deployment
        int version =
Integer.parseInt(info.get("RegisterAssemblyArchiveResult.version"));
        return version;
      } else {
        return -1;
      }

    } catch (Exception e) {
      Map<String, String> info = handleException(conn, e);
      for (Map.Entry<String, String> entry : info.entrySet()) {
        message("    "+entry.getKey()+"="+entry.getValue());
      }
      return -1;
    }

  }

  /*
   * This method shows how to wait for the completion of an asynchronous Deployer
request.
   * This is used by the examples for registration and deployment of an assembly
archive.
   */
  public static boolean waitForRequest(String id) throws Exception {

    String status = "RUNNING";

    while ("RUNNING".equals(status)) {
      message("    Async request "+id+" is still in progress");

      // Set up connection
      String params =
        "Action=DescribeRequests"+
        "&request.id="+id;

      HttpURLConnection conn = getConnection(DEPLOYER_
URL+"/ovab/deployer?"+params);
      conn.setDoInput(true);
```

```
      conn.setRequestMethod("GET");
      conn.setRequestProperty("Content-Type",
"application/x-www-form-urlencoded");
      conn.setRequestProperty("Content-Length", "" + params.length());

      // Use Application Admin Credentials
      sun.misc.BASE64Encoder enc = new sun.misc.BASE64Encoder();
      String encodedCredentials =
        enc.encode(new String(APPLICATION_ADMIN_USERNAME + ":" + APPLICATION_
ADMIN_PASSWORD).getBytes());
      conn.setRequestProperty("Authorization", "Basic " + encodedCredentials);

      // Make the request
      try {
        conn.connect();

        // Read the response
        Map<String, String> info = handleSuccessResponse(conn);
        status = info.get("DescribeRequestsResult.requestInfo.requestStatus");

        try { Thread.sleep(30*1000); } catch (InterruptedException e) { }

      } catch (Exception e) {
        Map<String, String> info = handleException(conn, e);
        for (Map.Entry<String, String> entry : info.entrySet()) {
          message("    "+entry.getKey()+"="+entry.getValue());
        }
        return false;
      }
    }

    message("    Request status is "+status);
    if ("SUCCEEDED".equals(status)) {
      return true;
    } else {
      return false;
    }
  }

  /*
   * This is an example of creating an assembly instance. This is a POST
   * operation because the deployment plan is supplied in this step.
   */
  public static String createAssemblyInstance(int version) throws Exception {
    message("CREATE Deployment");
    // Set up connection

    String urlStr =
      DEPLOYER_URL+
      "/ovab/deployer?"+
      "Action=CreateAssemblyInstance&assembly.name="+ASSEMBLY_NAME+
      "&assembly.version="+version+
      "&target="+TARGET_NAME;

    HttpURLConnection conn =
      getConnection(urlStr);

    conn.setDoInput(true);
    conn.setDoOutput(true);
    conn.setRequestMethod("POST");
```

```
    conn.setRequestProperty("Content-Type", "application/octet-stream");
    conn.setChunkedStreamingMode(0);

    sun.misc.BASE64Encoder enc = new sun.misc.BASE64Encoder();
    String encodedCredentials =
      enc.encode(new String(APPLICATION_ADMIN_USERNAME + ":" + APPLICATION_ADMIN_
PASSWORD).getBytes());
    conn.setRequestProperty("Authorization", "Basic " + encodedCredentials);

    // Make the request
    try {
      conn.connect();

      // Read plan document from a file and write to the web service
      int bytesRead, bufferSize;
      int maxBufferSize = 1 * 1024 * 1024;
      byte[] buffer;
      File f = new File(PLAN_FILE);
      FileInputStream fis = new FileInputStream(f);
      bufferSize = Math.min(fis.available(), maxBufferSize);
      buffer = new byte[bufferSize];
      bytesRead = fis.read(buffer, 0, bufferSize);
      OutputStream os = conn.getOutputStream();
      while (bytesRead > 0) {
        os.write(buffer, 0, bufferSize);
        bufferSize = Math.min(fis.available(), maxBufferSize);
        bytesRead = fis.read(buffer, 0, bufferSize);
      }
      fis.close();
      os.flush();
      os.close();

      // Read the response
      Map<String, String> info = handleSuccessResponse(conn);
      message("Deployment Created");
      for (Map.Entry<String, String> entry : info.entrySet()) {
        message("    "+entry.getKey()+"="+entry.getValue());
      }

      String assemblyInstanceId =
info.get("CreateAssemblyInstanceResult.assemblyInstanceId");
      return assemblyInstanceId;

    } catch (Exception e) {
      Map<String, String> info = handleException(conn, e);
      for (Map.Entry<String, String> entry : info.entrySet()) {
        message("    "+entry.getKey()+"="+entry.getValue());
      }

    }

    return null;

  }

  /*
   * This is an example of how to deploy an Assembly Instance
   *
   * Deployment is an asynchronous task, so this method calls
   * another method to wait for the asynchronous request to complete
```

```
   * before returning.
   *
   */
  public static boolean deployAssemblyInstance(String assemblyInstanceId) throws
Exception {

    message("Deploy Assembly Instance");

    // Set up connection
    String params =
      "Action=DeployAssemblyInstance"+
      "&assembly.instance.id="+assemblyInstanceId;
    HttpURLConnection conn = getConnection(DEPLOYER_URL+"/ovab/deployer?"+params);
    conn.setDoInput(true);
    conn.setRequestMethod("GET");
    conn.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
    conn.setRequestProperty("Content-Length", "" + params.length());

    // Use Application Admin Credentials
    sun.misc.BASE64Encoder enc = new sun.misc.BASE64Encoder();
    String encodedCredentials =
      enc.encode(new String(APPLICATION_ADMIN_USERNAME + ":" + APPLICATION_ADMIN_
PASSWORD).getBytes());
    conn.setRequestProperty("Authorization", "Basic " + encodedCredentials);

    // Make the request
    try {
      conn.connect();

      // Read the response
      message("    Async Deployment Request Initiated");
      Map<String, String> info = handleSuccessResponse(conn);
      for (Map.Entry<String, String> entry : info.entrySet()) {
        message("    "+entry.getKey()+"="+entry.getValue());
      }

      // Wait for the deployment to complete
      String requestId = info.get("DeployAssemblyInstanceResult.requestId");
      return waitForRequest(requestId);

    } catch (Exception e) {
      Map<String, String> info = handleException(conn, e);
      for (Map.Entry<String, String> entry : info.entrySet()) {
        message("    "+entry.getKey()+"="+entry.getValue());
      }
      return false;
    }

  }


  /*
   * This is a convenient method for creating an SSL connection to the Deployer
   */
  public static HttpURLConnection getConnection(String urlString) throws Exception
{
    URL url = new URL(urlString);

    if (false) {
      return (HttpURLConnection) url.openConnection();
```

```
    }

    // Need to setup SSL connection
    SocketFactory sf = null;
    final String ALGORITHM = "sunx509";
    // For Testing
    // For testing
    String pw = TRUSTSTORE_PASSWORD;
    final char[] keystorePass = pw.toCharArray();
    final char[] truststorePass = pw.toCharArray();

    // create the private keyStore
    KeyStore ksPrivate;
    ksPrivate = KeyStore.getInstance("JKS");
    ksPrivate.load(new FileInputStream(KEYSTORE_FILE), keystorePass);

    // create the factory for the private key
    KeyManagerFactory keyManagerFactory;
    keyManagerFactory = KeyManagerFactory.getInstance(ALGORITHM);
    keyManagerFactory.init(ksPrivate, keystorePass);

    // create now the trusted keyStore
    KeyStore ksTrusted;
    ksTrusted = KeyStore.getInstance("JKS");
    ksTrusted.load(new FileInputStream(TRUSTSTORE_FILE), truststorePass);

    // create a trust manager factory using the same algorithm.
    TrustManagerFactory trustManagerFactory;
    trustManagerFactory = TrustManagerFactory.getInstance(ALGORITHM);
    trustManagerFactory.init(ksTrusted);

    SSLContext sslc = SSLContext.getInstance("TLS");
    sslc.init(keyManagerFactory.getKeyManagers(),
trustManagerFactory.getTrustManagers(), null);

    sf = sslc.getSocketFactory();
    HttpsURLConnection httpsURLConn = (HttpsURLConnection)url.openConnection();
    httpsURLConn.setSSLSocketFactory((SSLSocketFactory) sf);

    httpsURLConn.setHostnameVerifier(new HostnameVerifier() {
        public boolean verify(String host, SSLSession sslsession) {
          return true;
        }
      });
    return (HttpURLConnection)httpsURLConn;

  }

  /*
   * This is a convenience method for extracting the information in the
   * response from a failed Deployer Web service call.
   */
  public static Map<String,String> handleException(HttpURLConnection conn,
Exception e) throws IOException, SAXException {

    Map<String,String> result = new HashMap<String,String>();

    message("ERROR: "+e.getMessage());
    message("RESPONSE CODE: "+conn.getResponseCode());
    if (conn.getResponseCode() == (HttpURLConnection.HTTP_NOT_FOUND)) {
```

```
      message("Invalid request, web service not found");
      return result;
    } else if (conn.getResponseCode() == (HttpURLConnection.HTTP_UNAUTHORIZED)) {
      message("User is not authorized to perform this action");
      return result;
    } else if (conn.getResponseCode() == (HttpURLConnection.HTTP_FORBIDDEN)) {
      message("User is not authorized to perform this action");
      return result;
    }

    // Read the response
    DataInputStream input = new DataInputStream(conn.getErrorStream());
    String str;
    StringBuffer response = new StringBuffer();
    try {
      while (null != ((str = input.readLine()))) {
        response.append(str);
      }
      input.close();

      // Parse the response
      XMLReader reader = XMLReaderFactory.createXMLReader();
      reader.setContentHandler(new ResponseHandler(result));
      reader.parse(new InputSource(new StringReader(response.toString())));

    } catch (Exception e2) {
      message("Failure reading error stream: "+e2);
    }

    return result;
  }

  /*
   * This is a convenience method for extracting the information in the
   * response from a successful Deployer Web service call.
   */
  public static Map<String,String> handleSuccessResponse(HttpURLConnection conn)
throws IOException, SAXException {
    HashMap<String,String> result = new HashMap<String,String>();
    DataInputStream input = new DataInputStream(conn.getInputStream());
    String str;
    StringBuffer response = new StringBuffer();
    while (null != ((str = input.readLine()))) {
      response.append(str);
    }
    input.close();

    // Parse the response
    XMLReader reader = XMLReaderFactory.createXMLReader();
    reader.setContentHandler(new ResponseHandler(result));
    reader.parse(new InputSource(new StringReader(response.toString())));

    return result;
  }

  /*
   * The responses returned from the Deployer Web service are XML
   * documents. This handler translates the structure of the document
   * into a Map which is useful for picking out certain element.  It
   * also prints the entries to System.out.
```

```
   */
  public static class ResponseHandler extends DefaultHandler {

    private Map<String,String> info;

    public ResponseHandler(Map<String,String> info) {
      this.info = info;
    }

    LinkedList<String> elements = new LinkedList<String>();

    public void startElement(String namespaceURI, String elementName, String
qName, Attributes attrs) {
      elements.add(elementName);
    }

    public void characters(char ch[], int start, int length) {
      String str = new String(ch,start,length);

      //message(String.format("    %s=%s", getCurrentKey(), str));
      info.put(getCurrentKey(), str);
    }

    private String getCurrentKey() {
      String result = null;
      for (String s : elements) {
        if (result != null) {
          result = result + "." +s;
        } else {
          result = s;
        }
      }
      return result;
    }

    public void endElement(String namespaceURI, String elementName, String qName)
{
      elements.removeLast();
    }
  }

  private static void message(String msg) {
    System.out.println("["+new Date(System.currentTimeMillis())+"] "+msg);
 }

}
```

A

Web Service Schema

This appendix defines the schema for the response information returned by the Deployer Web service. The types defined in this schema are referenced by the "Response Content" descriptions in earlier chapters of the guide.

A.1 Schema

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.oracle.com/ovab/deployer/webservice/bindings"
    xmlns:tns="http://www.oracle.com/ovab/deployer/webservice/bindings"
    elementFormDefault="qualified">
    <xsd:element name="Path">
        <xsd:complexType>
            <xsd:attribute name="value" type="xsd:string"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="InstancePath">
        <xsd:complexType>
            <xsd:attribute name="value" type="xsd:string"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="AppliancePath">
        <xsd:complexType>
            <xsd:attribute name="value" type="xsd:string"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="Paths">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="tns:Path" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="AssemblyInstanceId">
        <xsd:complexType>
            <xsd:attribute name="value" type="xsd:string"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="AssemblyInstanceState">
        <xsd:complexType>
            <xsd:attribute name="id" type="xsd:string"/>
            <xsd:attribute name="state" type="xsd:string"/>
```

```
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="UploadAssemblyArchiveResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="name" type="tns:nameType"></xsd:element>
                <xsd:element name="version" type="tns:versionType"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="DeleteAssemblyArchiveResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="name" type="tns:nameType"></xsd:element>
                <xsd:element name="versions"
type="tns:VersionsType"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="ErrorResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="code">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string"/>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="type">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string"/>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="message">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string"/>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="cause">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string"/>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="action">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string"/>
                    </xsd:simpleType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:complexType name="VersionsType">
        <xsd:sequence>
            <xsd:element name="version" maxOccurs="unbounded"
type="tns:versionType"></xsd:element>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:element name="RegisterAssemblyArchiveResult">
        <xsd:complexType>
            <xsd:sequence>
```

```
                <xsd:element name="requestId"
type="tns:requestIdType"></xsd:element>
                <xsd:element name="name" type="tns:nameType"></xsd:element>
                <xsd:element name="version" type="tns:versionType"></xsd:element>
                <xsd:element name="targetName"
type="tns:targetNameType"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="UnregisterAssemblyArchiveResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="requestId"
type="tns:requestIdType"></xsd:element>
                <xsd:element name="name" type="tns:nameType"></xsd:element>
                <xsd:element name="version" type="tns:versionType"></xsd:element>
                <xsd:element name="targetName"
type="tns:targetNameType"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:simpleType name="requestIdType">
        <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>
    <xsd:simpleType name="nameType">
        <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>
    <xsd:simpleType name="versionType">
        <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>
    <xsd:element name="DescribeAssemblyArchivesResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="assembly" type="tns:Assembly"
maxOccurs="unbounded"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:complexType name="AssemblyVersion">
        <xsd:sequence>
            <xsd:element name="version" type="tns:versionType"></xsd:element>
            <xsd:element name="description"
type="tns:descriptionType"></xsd:element>
            <xsd:element name="appliances">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="appliance" type="xsd:string"
maxOccurs="unbounded"></xsd:element>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="Assembly">
        <xsd:sequence>
            <xsd:element name="name" type="tns:nameType"></xsd:element>
            <xsd:element name="versions">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="version" type="tns:AssemblyVersion"
```

```
maxOccurs="unbounded"></xsd:element>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="plans">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="plan" maxOccurs="unbounded"
type="xsd:string"></xsd:element>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:simpleType name="descriptionType">
        <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>
    <xsd:element name="UpdateAssemblyArchiveResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="name" type="tns:nameType"></xsd:element>
                <xsd:element name="version" type="tns:versionType"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:simpleType name="assemblyInstanceIdType">
        <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>
    <xsd:element name="CreateAssemblyInstanceResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="assemblyInstanceId"
type="tns:assemblyInstanceIdType"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="DeleteAssemblyInstanceResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="assemblyInstanceId"
type="tns:assemblyInstanceIdType"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="DeployAssemblyInstanceResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="requestId"
type="tns:requestIdType"></xsd:element>
                <xsd:element name="assemblyInstanceId"
type="tns:assemblyInstanceIdType"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="UndeployAssemblyInstanceResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="requestId"
type="tns:requestIdType"></xsd:element>
                <xsd:element name="assemblyInstanceId"
```

```
type="tns:assemblyInstanceIdType"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="StartAssemblyInstanceResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="requestId"
type="tns:requestIdType"></xsd:element>
                <xsd:element name="assemblyInstanceId"
type="tns:assemblyInstanceIdType"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="StopAssemblyInstanceResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="requestId"
type="tns:requestIdType"></xsd:element>
                <xsd:element name="assemblyInstanceId"
type="tns:assemblyInstanceIdType"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:simpleType name="instanceIdType">
        <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>
    <xsd:complexType name="InstancesType">
        <xsd:sequence>
            <xsd:element name="instanceId" type="tns:instanceIdType"
maxOccurs="unbounded"></xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:simpleType name="targetNameType">
        <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>
    <xsd:complexType name="Instance">
        <xsd:sequence>
            <xsd:element name="assemblyInstanceId"
type="xsd:string"></xsd:element>
            <xsd:element name="instanceId" type="xsd:string"></xsd:element>
            <xsd:element name="instanceState" type="xsd:string"></xsd:element>
            <xsd:element name="applianceId" type="xsd:string"></xsd:element>
            <xsd:element name="appliancePath" type="xsd:string"></xsd:element>
            <xsd:element name="vmName" type="xsd:string"></xsd:element>
            <xsd:element name="ipaddresses">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="ipaddress" maxOccurs="unbounded"
type="xsd:string"></xsd:element>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="vnets">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="vnet" maxOccurs="unbounded"
type="tns:Vnet"></xsd:element>
                    </xsd:sequence>
                </xsd:complexType>
```

```
            </xsd:element>
            <xsd:element name="volumes">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="volume" maxOccurs="unbounded"
type="xsd:string"></xsd:element>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="meteringMap">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="metering" maxOccurs="unbounded">
                            <xsd:complexType>
                                <xsd:sequence>
                                    <xsd:element name="name"
type="xsd:string"></xsd:element>
                                    <xsd:element name="value"
type="xsd:long"></xsd:element>
                                </xsd:sequence>
                            </xsd:complexType>
                        </xsd:element>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="AssemblyInstance">
        <xsd:sequence>
            <xsd:element name="assemblyInstanceId"
type="xsd:string"></xsd:element>
            <xsd:element name="assemblyName" type="xsd:string"></xsd:element>
            <xsd:element name="assemblyVersion" type="xsd:string"></xsd:element>
            <xsd:element name="state" type="xsd:string"></xsd:element>
            <xsd:element name="targetName" type="xsd:string"></xsd:element>
            <xsd:element name="appliances">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="appliance" type="xsd:string"
maxOccurs="unbounded"></xsd:element>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="DescribeAssemblyInstancesResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="assemblyInstance" type="tns:AssemblyInstance"
maxOccurs="unbounded"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:complexType name="RequestInfo">
        <xsd:sequence>
            <xsd:element name="requestId" type="tns:requestIdType"></xsd:element>
            <xsd:element name="requestStatus" type="xsd:string"></xsd:element>
            <xsd:element name="requestType" type="xsd:string"></xsd:element>
            <xsd:element name="operationType" type="xsd:string"></xsd:element>
            <xsd:element name="assemblyName" type="xsd:string"></xsd:element>
```

```
            <xsd:element name="message" type="xsd:string"></xsd:element>
            <xsd:element name="creationTimeStamp" type="xsd:long"></xsd:element>
            <xsd:element name="completionTimeStamp" type="xsd:long"></xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="DescribeRequestsResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="requestInfo" type="tns:RequestInfo"
maxOccurs="unbounded"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="DescribeDeployerResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="specificationVersion"
type="xsd:string"></xsd:element>
                <xsd:element name="implementationVersion"
type="xsd:string"></xsd:element>
                <xsd:element name="assemblyStoreFreeSpace"
type="xsd:long"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="CreateTargetResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="target" type="xsd:string"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="DescribeTargetNamesResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="target" type="xsd:string"
maxOccurs="unbounded"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="DeleteTargetResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="target" type="xsd:string"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="SetDefaultTargetResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="targetName" type="xsd:string"></xsd:element>
                <xsd:element name="targetType" type="xsd:string"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="DescribeApplianceInstancesResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="instance" type="tns:Instance"
maxOccurs="unbounded"></xsd:element>
```

```
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="AddTargetUserResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="target" type="xsd:string"></xsd:element>
                <xsd:element name="user" type="xsd:string"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="RemoveTargetUsersResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="target" type="xsd:string"></xsd:element>
                <xsd:element name="users">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="user" type="xsd:string"
maxOccurs="unbounded"></xsd:element>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:complexType name="Properties">
        <xsd:sequence>
            <xsd:element name="entry" minOccurs="0" maxOccurs="unbounded">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="key" minOccurs="1" maxOccurs="1"
type="xsd:string"/>
                        <xsd:element name="value" minOccurs="1" maxOccurs="1"
type="xsd:string"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="TargetUser">
        <xsd:sequence>
            <xsd:element name="userName" type="xsd:string"></xsd:element>
            <xsd:element name="targetProperties"
type="tns:Properties"></xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="DescribeTargetUsersResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="user" type="tns:TargetUser"
maxOccurs="unbounded"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="DescribeUserTargetsResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="target" type="xsd:string"
maxOccurs="unbounded"></xsd:element>
```

```
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="GetTargetTypeResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="targetType" type="xsd:string"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="GetDefaultTargetResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="targetName" type="xsd:string"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:complexType name="ScalingGroup">
        <xsd:sequence>
            <xsd:element name="scalingGroupId" type="xsd:string"></xsd:element>
            <xsd:element name="min" type="xsd:int"></xsd:element>
            <xsd:element name="max" type="xsd:int"></xsd:element>
            <xsd:element name="target" type="xsd:int"></xsd:element>
            <xsd:element name="current" type="xsd:int"></xsd:element>
            <xsd:element name="initial" type="xsd:int"></xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="DescribeScalingGroupsResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="scalingGroups" type="tns:ScalingGroup"
minOccurs="0" maxOccurs="unbounded"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="ScaleApplianceResult">
       <xsd:complexType>
        <xsd:sequence>
       <xsd:element name="requestId" type="tns:requestIdType"></xsd:element>
            <xsd:element name="scalingGroupId" type="xsd:string"></xsd:element>
            <xsd:element name="assemblyInstanceId"
type="xsd:string"></xsd:element>
        </xsd:sequence>
       </xsd:complexType>
    </xsd:element>
    <xsd:complexType name="Vnet">
       <xsd:sequence>
           <xsd:element name="vnetId" type="xsd:string"></xsd:element>
           <xsd:element name="vnetName" type="xsd:string"></xsd:element>
           <xsd:element name="netmask" type="xsd:string"></xsd:element>
           <xsd:element name="private" type="xsd:boolean"></xsd:element>
       </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="DescribeVnetsResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="vnet" type="tns:Vnet" minOccurs="0"
maxOccurs="unbounded"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
```

```
    </xsd:element>
    <xsd:complexType name="Target">
        <xsd:sequence>
            <xsd:element name="targetName"
type="tns:targetNameType"></xsd:element>
            <xsd:element name="targetType" type="xsd:string"></xsd:element>
            <xsd:element name="status" type="xsd:string"></xsd:element>
            <xsd:element name="vnetNames">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="vnetName" maxOccurs="unbounded"
type="xsd:string"></xsd:element>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="volumeNames">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="volumeName" maxOccurs="unbounded"
type="xsd:string"></xsd:element>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="availabilityMap">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="metering" maxOccurs="unbounded">
                            <xsd:complexType>
                                <xsd:sequence>
                                    <xsd:element name="name"
type="xsd:string"></xsd:element>
                                    <xsd:element name="value"
type="xsd:long"></xsd:element>
                                </xsd:sequence>
                            </xsd:complexType>
                        </xsd:element>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="DescribeTargetsResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="target" type="tns:Target" minOccurs="0"
maxOccurs="unbounded"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:complexType name="RegistrationInfo">
        <xsd:sequence>
            <xsd:element name="assemblyName" type="xsd:string"></xsd:element>
            <xsd:element name="assemblyVersion" type="xsd:string"></xsd:element>
            <xsd:element name="targetName" type="xsd:string"></xsd:element>
            <xsd:element name="registrationId" type="xsd:string"></xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="DescribeRegistrationsResult">
        <xsd:complexType>
            <xsd:sequence>
```

```
                <xsd:element name="registration" type="tns:RegistrationInfo"
minOccurs="0" maxOccurs="unbounded"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:complexType name="TargetConfiguration">
        <xsd:sequence>
            <xsd:element name="targetName" type="xsd:string"></xsd:element>
            <xsd:element name="targetType" type="xsd:string"></xsd:element>
            <xsd:element name="properties">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="property" maxOccurs="unbounded">
                            <xsd:complexType>
                                <xsd:sequence>
                                    <xsd:element name="name"
type="xsd:string"></xsd:element>
                                    <xsd:element name="value"
type="xsd:string"></xsd:element>
                                </xsd:sequence>
                            </xsd:complexType>
                        </xsd:element>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="DescribeTargetConfigurationsResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="targetConfiguration"
type="tns:TargetConfiguration" minOccurs="0" maxOccurs="unbounded"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="RestartAssemblyInstanceResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="requestId"
type="tns:requestIdType"></xsd:element>
                <xsd:element name="assemblyInstanceId"
type="tns:assemblyInstanceIdType"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="RedeployAssemblyInstanceResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="requestId"
type="tns:requestIdType"></xsd:element>
                <xsd:element name="assemblyInstanceId"
type="tns:assemblyInstanceIdType"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="DeleteRequestsResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="requestsPurged"
type="xsd:string"></xsd:element>
```

```
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="DescribeAssemblyUsersResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="assemblyName" type="xsd:string"></xsd:element>
                <xsd:element name="users" type="xsd:string"
maxOccurs="unbounded"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="AddAssemblyUsersResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="assemblyName" type="xsd:string"></xsd:element>
                <xsd:element name="users" type="xsd:string"
maxOccurs="unbounded"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="RemoveAssemblyUsersResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="assemblyName" type="xsd:string"></xsd:element>
                <xsd:element name="users" type="xsd:string"
maxOccurs="unbounded"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="CreateTagsResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="resources" type="xsd:string"
maxOccurs="unbounded"></xsd:element>
                <xsd:element name="tags">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="tag" maxOccurs="unbounded">
                                <xsd:complexType>
                                    <xsd:sequence>
                                        <xsd:element name="key"
type="xsd:string"></xsd:element>
                                        <xsd:element name="value"
type="xsd:string"></xsd:element>
                                    </xsd:sequence>
                                </xsd:complexType>
                            </xsd:element>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="DeleteTagsResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="resources" type="xsd:string"
maxOccurs="unbounded"></xsd:element>
                <xsd:element name="tags" type="xsd:string"
```

```
maxOccurs="unbounded"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="DescribeTagsResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="tags">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="tag" maxOccurs="unbounded">
                                <xsd:complexType>
                                    <xsd:sequence>
                                        <xsd:element name="key"
type="xsd:string"></xsd:element>
                                        <xsd:element name="value"
type="xsd:string"></xsd:element>
                                    </xsd:sequence>
                                </xsd:complexType>
                            </xsd:element>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="UpdateTargetResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="targetName" type="xsd:string"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="DeleteFailedApplianceInstancesResult">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="applianceId" type="xsd:string"></xsd:element>
                <xsd:element name="applianceInstanceIds" type="xsd:string"
maxOccurs="unbounded"></xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```