Oracle® Fusion Middleware

Using Oracle Virtual Assembly Builder

12*c* (12.1.2)

E29476-02

August 2013

Documentation for users and system administrators that describes how to use and configure Oracle Virtual Assembly Builder.

ORACLE

Oracle Fusion Middleware Using Oracle Virtual Assembly Builder, 12*c* (12.1.2)

E29476-02

Contents

B Oracle Virtual Assembly Builder Introspection Plug-ins

C Common Properties for Oracle Virtual Assembly Builder Components

D Troubleshooting

E Third-Party Licensing

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 system administrators who will use Oracle Virtual Assembly Builder for their organizations.

What's New in This Release

This release of Oracle Virtual Assembly Builder contains the following new features or product enhancements:

- The latest versions of introspection plug-ins are shipped with their respective Oracle Fusion Middleware components and installed into those component's Oracle Home at installation time. Oracle Virtual Assembly Builder can discover those plug-ins and install them into the Oracle Virtual Assembly Builder installation for use in introspection.
- Ability to customize introspection plug-ins, for example to expose a specific parameter as a late binding, or execute a custom script on reconfiguration. Changes to the scripting API that supports scripts for the custom reconfiguration scripts and generic appliances features. Lifecycle enhancements for the Deployer affecting reconfiguration scripts.
- Custom appliance properties: Ability to add custom properties to an appliance that can be edited along with appliance's pre-defined properties during assembly editing and as part of deployment plan. With this feature you can configure, and/or operate a custom product or component that gets deployed with an Oracle product in an appliance.
- Support for configuring base OS images used to create an appliance with logical volumes (Logical Volume Management).

- Wiring support for generic appliances: you can now perform wiring of generic appliances to connect from/to other appliances in the assembly, connect to external resources, and exchange properties between connected appliances.
- Enhancements and new features in Oracle Virtual Assembly Builder Studio, as well as new `abctl` command-line and Web service interfaces:
  - Commands to check assembly requirements against available resources.
  - Deployer connection manager for managing connections to the Oracle Virtual Assembly Builder Deployer.
  - A new Task viewer tracks long running operations and allows cancellation of some operations.
- Deployer as a clustered service: the Deployer can run in an Oracle WebLogic Server cluster with two or more managed servers and take advantage of the clustering to balance the work amongst multiple servers and avoid a single point of failure.

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:

- *Installing Oracle Virtual Assembly Builder*
- *Developing Applications and Introspection Plug-ins for Oracle Virtual Assembly Builder*
- *Release Notes for Oracle Virtual Assembly Builder*

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 Introduction

This chapter provides an introduction to the major concepts and components of Oracle Virtual Assembly Builder, and contains the following sections:

1.1 Introduction to Oracle Virtual Assembly Builder

Increased operating costs, inefficient hardware utilization and rapidly expanding data centers have made virtualization the most compelling IT technology in years. Virtualization for desktop and server environments has evolved to finally deliver on its promise to lower operating costs by increasing the utilization of hardware and reducing the overall amount of hardware required.

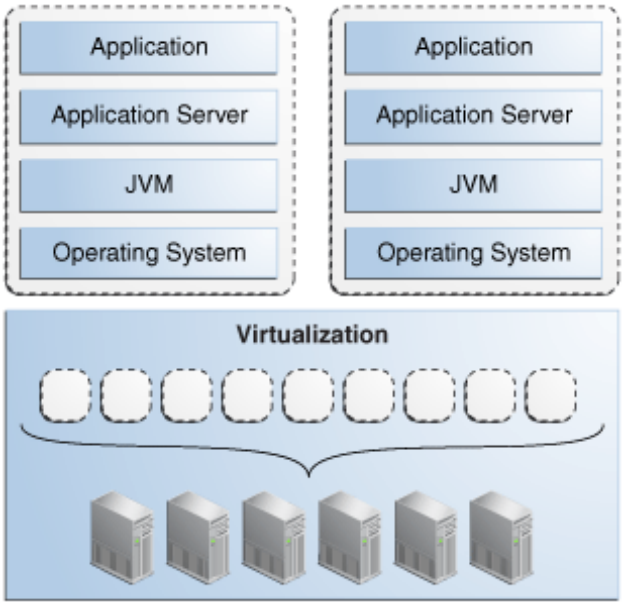
While virtualization has solved a multitude of problems, it is still difficult to deploy and manage complex applications made up of multiple tiers and components. Furthermore, virtualization is quickly becoming a commodity and the focus now shifts to directly virtualizing applications to reap the next level of benefits associated with virtualization.

1.1.1 What is Virtualization?

Virtualization is the process of abstracting hardware resources, such as CPU, memory, storage, and network interfaces, from the operating system and applications. The hardware runs virtualization software (for example, a hypervisor) that enables the installation of multiple operating systems, each capable of running simultaneously and independently, in its own secure physical environment.

The goal of virtualization is to make deployment of complete environments faster, easier, and more efficient. Virtualization's capabilities must be integrated to facilitate deployment and management of complete stacks. Virtualization must enable the entire stack to be easier to deploy, manage, and support.

Figure 1–1 Virtualization



1.1.2 Middleware Virtualization Challenges

The development and deployment of applications in your virtualized environment involves a sequence of operational stages including testing, staging, and production. The transition between these stages can be difficult as there are few facilities within existing virtualization infrastructure that guarantee consistency and correctness of the collection of software components. Implementing the physical to virtual (P2V) or virtual to virtual (V2V) transitions seems simple: create virtual images of the original deployments, then instantiate them in the target environment. Oracle VM can be used to implement such solutions.

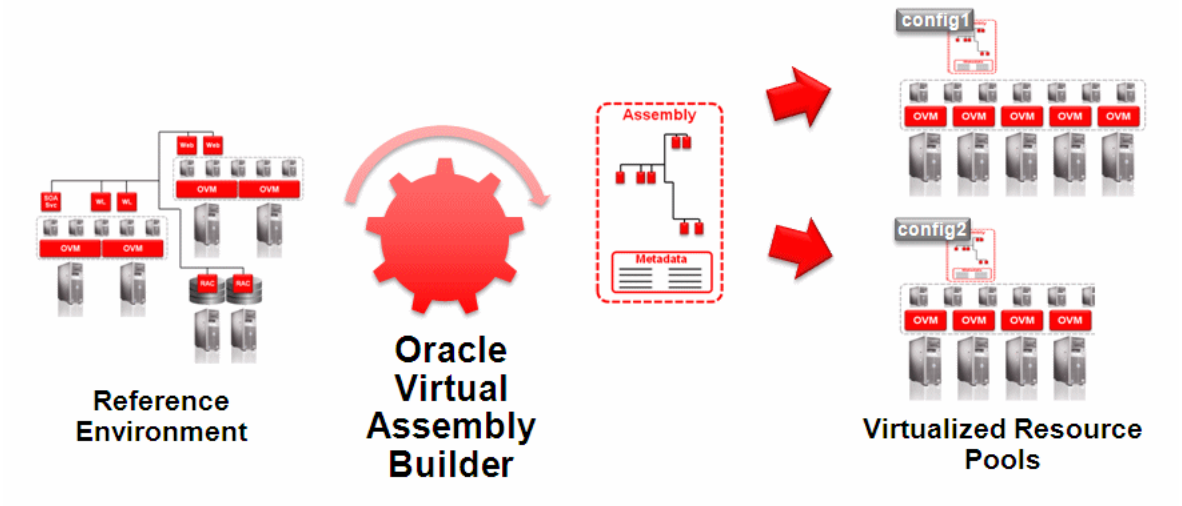
Handcrafting the virtualization solution has many pitfalls. Details of network connectivity may change in the deployment environment, but no automatic mechanism exists to perform or even to track these changes. Images may be specific to particular details of the deployment environment. The proliferation of images results in *sprawl*, creating maintenance overhead as each of the images must be patched at the operating system and application layers. These pitfalls create unanticipated costs.

1.1.3 What is Oracle Virtual Assembly Builder?

Oracle Virtual Assembly Builder is a tool for virtualizing installed Oracle components, modifying those components, and then deploying them into your own environment. Using Oracle Virtual Assembly Builder, you capture the configuration of existing software components in artifacts called software *appliances*. Appliances can then be grouped, and their relationships defined into artifacts called software *assemblies* which provide a blueprint describing a complete multi-tier application topology.

Oracle Virtual Assembly Builder allows the logical connections between appliances within an assembly to be reconfigured by a process known as *assembly editing*. When a desired assembly configuration has been achieved, you use Oracle Virtual Assembly Builder to prepare the assembly for deployment and then deploy it into your environment. The components and processes are described below.

Figure 1–2 Oracle Virtual Assembly Builder



1.1.4 Software Appliances

A software appliance (*appliance*) represents a single software component and its local execution environment.

1.1.5 Software Assemblies

A software assembly (*assembly*) is a collection of interrelated software appliances that are automatically configured to work together upon deployment. Assemblies are deployed onto a pool of hardware resources with minimal user input.

While assemblies are simply a collection of appliances with defined interconnects, assemblies must provide a set of capabilities in order to be useful in a production environment, including:

- Allow for the composition of appliances as well as external systems
- Externalize configuration in the form of metadata that can easily be customized
- Optionally define the start order of appliances to reflect interdependencies
- Provide a management domain which integrates into existing management infrastructure allowing for metadata definition, deployment, oversight and diagnostics

In addition to being comprised of appliances, assemblies can also contain references to *external systems*. This is necessary to represent infrastructure such as databases, servers or security providers that cannot or should not be included in an assembly.

To summarize, the notion of being able to create pre-built assemblies for deployment is extremely powerful and has a number of advantages that drive down operational costs and complexity. These include:

- Ability to easily replicate assemblies in production, even allowing for variations of the assembly without adding complexity
- Reduced risk of configuration errors as assemblies are moved between development, test and production environments

- Replicated environments facilitate high-level standardization and consistency across application infrastructures, allowing for simple implementation of best practices.
- Accelerated deployment of new infrastructures and applications

1.1.6 The Role of Oracle Virtual Assembly Builder

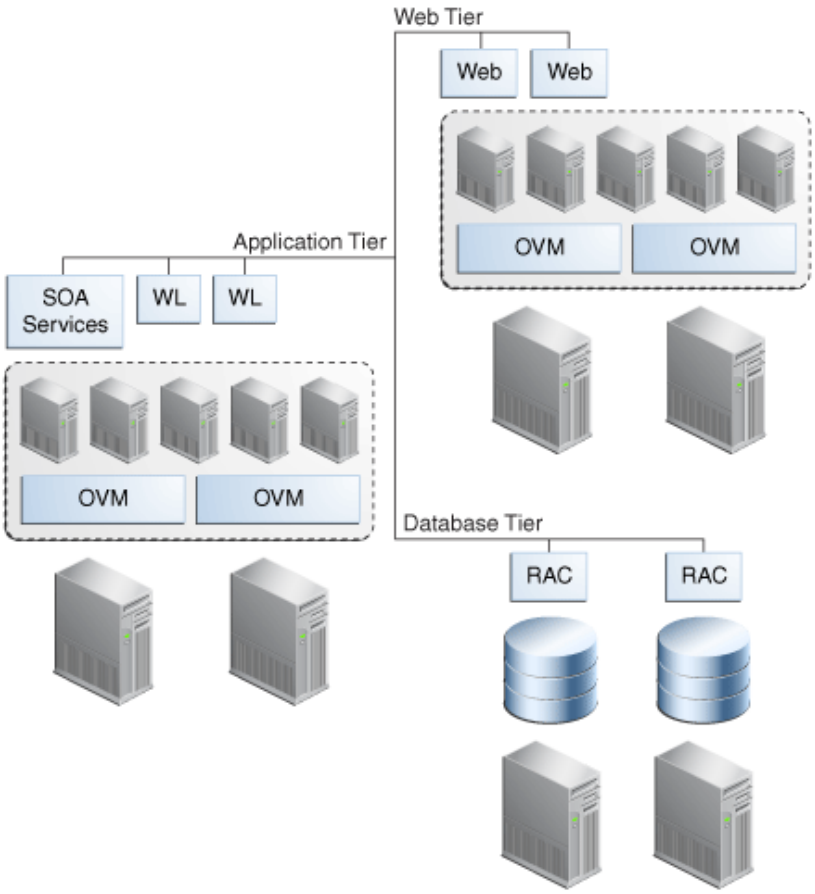
In order to realize these benefits, a simple means of composing assemblies of appliances is required. Specifically what is needed is tooling that allows for the composition of appliances as well as endpoint mapping of externalized systems and other larger non-virtual appliance-based systems such as databases and identity management servers.

Oracle Virtual Assembly Builder includes an intuitive visual environment, command line interface, and supporting infrastructure. Oracle Virtual Assembly Builder enables administrators to construct and deploy complete assemblies encompassing all of the components and systems that make up a potentially complex application structure or infrastructure.

Oracle Virtual Assembly Builder provides the following capabilities:

- Ability to browse a catalog of existing appliances and assemblies allowing for simple re-use of existing infrastructure
- Assembly editor that enables declarative composition of new assemblies based on existing appliances and external systems
- Ability to modify connections between appliances using drag-and-drop
- Property inspector that displays the editable properties of appliances and assemblies
- Ability to create templatized definitions of complete configurations, allowing for simple deployment
- Single-step deployment of virtualized multi-tier applications onto a pool of virtualized resources

Figure 1–3 Virtualized Multi-Tier Applications



1.2 Understanding Oracle Virtual Assembly Builder

Oracle Virtual Assembly Builder captures the existing condition of a specific set of Oracle Fusion Middleware and Oracle Database software components from your environment, represents them as assemblies and appliances, and enables their deployment into your environment. Oracle Virtual Assembly Builder does not include the ability to administer the components and does not replace the administrative tools supplied with them.

Oracle Virtual Assembly Builder does not supply the virtual environment into which you deploy your Assemblies. You must establish the deployment environment using one of the target environments that Oracle Virtual Assembly Builder supports. For more information about supported deployment environments, see *Installing Oracle Virtual Assembly Builder*.

1.2.1 Product Components

Oracle Virtual Assembly Builder consists of two major product components:

- Oracle Virtual Assembly Builder Studio provides you the capabilities to perform the first three phases of the assembly creation, the introspect phase, configure phase, and prepare phase. Oracle Virtual Assembly Builder Studio allows you to create and edit the assemblies, create assembly archives, and create templates and deployment plans which support deployment from Oracle Virtual Assembly Builder Deployer.

- Oracle Virtual Assembly Builder Deployer is a Java EE application that maintains a repository of assembly archives created by Oracle Virtual Assembly Builder Studio. The Deployer provides operations for registering these assembly archives to virtualized systems such as Oracle VM and provides operations for orchestrating the deployment of the software system defined by the assembly archive.

  The interface to Oracle Virtual Assembly Builder Deployer is a Web service which provides operations for uploading assembly archives, registering the assembly archive virtualization system and managing assembly instances for the system defined in the assembly archive. See *Developing Applications and Introspection Plug-ins for Oracle Virtual Assembly Builder* for a description of the Web service.

1.2.2 Appliances and Assemblies

A minimal appliance consists of metadata (name and value pairs) describing the condition of the original component, together with a set of component-specific files that allow its configuration to be recreated at deployment time. As you use Oracle Virtual Assembly Builder to prepare an assembly for deployment into your environment, additional configuration information is created and stored along with the metadata.

The appliance metadata includes a description of each of the component's logical inputs and outputs. These inputs and outputs are collectively called *endpoints*. The HTTP input of an Oracle HTTP Server component is an example of an input endpoint. The `mod_wl_ohs` output of the same Oracle HTTP Server component is an example of an output endpoint.

The metadata describing endpoints includes protocols, port numbers, URLs, and so on. Oracle Virtual Assembly Builder captures enough information about each endpoint to allow the connection to be updated after the component is captured and before it is deployed. This capability allows Oracle Virtual Assembly Builder to ensure that the appliances will connect correctly within the deployment environment.

Appliances are grouped into *assemblies*. An assembly is a logical container for the appliances and the connections between them. You create assemblies using Oracle Virtual Assembly Builder and populate them with the appliances and the other assemblies (assemblies may contain other assemblies).

The process of capturing a software component from your environment as an Oracle Virtual Assembly Builder appliance begins with *introspection*.

1.2.3 Introspection

Introspection is an operation performed on a software component or a group of related components (to create an appliance or assembly). During introspection, Oracle Virtual Assembly Builder creates an xml description of the component and captures a component-specific set of configuration files. This information forms a snapshot of the component's configuration at the time of introspection. The introspection architecture is plug-in based and there is a plug-in for each supported component type. See *Appendix B, Oracle Virtual Assembly Builder Introspection Plug-ins* for more information about available plug-ins.

In most cases, the result of introspecting a component is an appliance. When you use Oracle Virtual Assembly Builder to introspect a WebLogic domain; however, the Introspector plug-in generates an assembly. The generated assembly contains an appliance representing the domain's Administration Server and other appliances representing each of the domain's Managed Servers.

Oracle Virtual Assembly Builder can introspect components on a local host or components located on remote, network-accessible hosts. Oracle Virtual Assembly Builder uses the industry-standard SSH protocol to transport the introspection engine to the remote host and to return the introspection results.

Whether the introspection is local or remote, the results are stored locally in the catalog.

Users who want to create their own customized introspection plug-ins can use the Oracle Virtual Assembly Builder Introspection SDK. For details on creating your own plug-ins, see *Developing Applications and Introspection Plug-ins for Oracle Virtual Assembly Builder*.

1.2.4 Generic Appliances

Appliances constructed using the appliance type called "GenericProd". These type of appliances do not make use of product-specific logic to capture configuration or product location, instead a simple appliance is created and a set of user-supplied properties, paths, and scripts that make up the product are added to it in a generic manner. Also it does not make use of any product-specific logic to configure and start the product upon deployment, instead the set of scripts passed in at creation are executed at deployment to perform the necessary operations.

This allows user to create and deploy an opaque, standalone, and self-contained product or application as an appliance for which Oracle Virtual Assembly Builder does not have built-in support.

1.2.5 Catalog

Assemblies and appliances are represented on disk in an area called the *Catalog*. Assembly and appliance metadata is stored in nested directories within the metadata subfolder of the catalog root directory. Additional artifacts required for deployment are stored in other subdirectories defined by Oracle Virtual Assembly Builder. Since some of the on-disk artifacts may be very large, the catalog uses a sharing model for some artifacts of appliances and assemblies.

Only Oracle Virtual Assembly Builder-supplied tools should be used to operate on the catalog. Manually editing Oracle Virtual Assembly Builder metadata files is not supported.

1.2.6 External Resources

When defining an assembly, it may be necessary to make reference to servers that lie outside it. Your IT environment may, for example, include database, identity management, or other servers that are shared by many unrelated virtual deployments. It may be undesirable or impossible to include these systems within any specific assembly. For this reason, Oracle Virtual Assembly Builder enables you to define external components representing server resources that exist in your environment and will not be deployed as appliances. Representing them as external resources ensures that referencing appliance(s) within the assembly are correctly configured at deployment time, making it unnecessary to manually correct their network configuration after they are deployed to the virtual environment.

1.2.7 Capturing File Sets

The introspection process captures the condition of a component and generates a metadata description of the actual component installation. Introspection does not

capture the executables, shared libraries or other binaries of the component. Instead, introspection generates file set definitions that specify one or more file system hierarchies that must be captured to reproduce the same component installation in the deployment environment. By default, after the introspection is complete, Oracle Virtual Assembly Builder automatically captures a copy of the actual installation described by the metadata. This step is known as *capturing file sets*.

By default, introspection and capturing file sets are done together whether you use Oracle Virtual Assembly Builder Studio or Oracle Virtual Assembly Builder command line interface. Optionally, you can choose to do these steps separately.

1.2.7.1 Assembly Archive

The assembly archives created by Oracle Virtual Assembly Builder Studio contain information about a software system composed 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 assembly templates that are used to instantiate an instance of the assembly in a virtualized environment.

1.2.7.2 File Sets and Shared File Sets

You can configure a file set as shared or local. If supported by the underlying infrastructure platform, you can specify that file sets be shared with individual appliances within assemblies.

1.2.7.3 Networks

The assembly archive defines a set of logical networks for the application it represents. For each appliance, the assembly archive also defines one or more network interfaces. Each network interface is associated with one of the archive's networks, allowing the assembly archive to fully represent the network connectivity requirements of the assembly.

The deployment plan specifies a network in the virtualization environment to be used for each logical network declared in the assembly archive. The model supports the configuration and binding to both public and private networks (where private is defined as existing between two appliances in an assembly and not surfaced as part of the public network for access to the deployed application topology.)

The Deployer creates and attaches one or more Vnets to the virtual machines it creates using the underlying interfaces of the virtualization system. Note that these Vnets are hypervisor-level Vnets, as opposed to virtual machine-level Vnets. If the virtualization system supports it, the Deployer may also dynamically create private networks to associate these Vnets with.

1.2.8 Assembly Templates

An assembly template is a set of virtual disk images that can be used to create and start new virtual machine instances. A template is created for each appliance in an assembly, consisting of a guest operating system, the appliance's file sets and metadata, and supporting Oracle Virtual Assembly Builder infrastructure. Templates are made available to the virtualized environment by registering them to that environment, at which point virtual machine instances can be created based on the templates.

Oracle Virtual Assembly Builder supports Oracle Enterprise Linux as the virtual machine guest operating system.

1.2.9 Deployment Plans

Deployment plans are used to customize assemblies prior to deployment. You can create a deployment plan in which you customize default assembly and appliance properties, and provide deployment-specific information such as network configuration. In some cases you may be required to customize certain properties in order to proceed with a successful deployment (for example, static IP address, or password properties).

1.2.10 Using Oracle Virtual Assembly Builder

Assembly creation and deployment is a straightforward, four-step process. First, in the *introspect* phase, the necessary metadata and configuration information is captured from an existing deployment for all components that make up the appliances within an assembly. During the *configure* phase, the relationships are established among the appliances and any external resources. The *prepare* phase creates the deployment artifacts necessary for the assembly that is relevant to the particular virtualization platform (that is, virtual images). Finally, the *deploy* phase deploys the assembly into your environment.

1.2.10.1 Introspect

In the introspect phase, you capture configuration metadata for individual software components, or collectively capture metadata for multiple distributed components. Target components may reside locally or remotely on multiple distributed systems that may be physical or virtual.

1.2.10.2 Configure

In the configure phase, you:

- Visually drag-and-drop components for creating complex assemblies using appliances maintained in a navigable catalog
- Establish relationships and connections between appliances using a wiring tool that automatically checks for protocol compatibility
- Create connections from appliances to external resources (such as database, security provider, messaging, and so on) not included within the assembly

1.2.10.3 Prepare

In the prepare phase, you:

- Create bootable virtual machine disk images with customized Oracle Enterprise Linux operating system distributions and configurable metadata allowing for deploy-time customization of the software component

1.2.10.4 Deploy

In the deploy phase, you:

- *Discover* targets available on virtualized environments by establishing authenticated connections directly with a virtual machine manager
- Create customized deployment configurations for assemblies that override base configuration properties for the appliances within the assembly
- Stage all appliance disk images and deploy entire assemblies onto targets in a single step

- Scale appliance instances after initial deployment of an assembly and perform assembly lifecycle operations such as scale in/out, start, and stop

1.2.11 Understanding Deployer Concepts

This section describes Deployer concepts.

1.2.11.1 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.

1.2.11.2 Assembly Instances

An assembly instance is a deployable instance of an assembly archive for a specific target virtual environment.

1.2.11.3 Appliance Instances

An appliance instance is an instance of an appliance running and/or created in the target virtual environment.

After you deploy an assembly instance, the target number of appliance instances for each appliance is started. The initial target for each appliance is specified in the deployment plan. You can dynamically specify a new target after an assembly instance has been deployed.

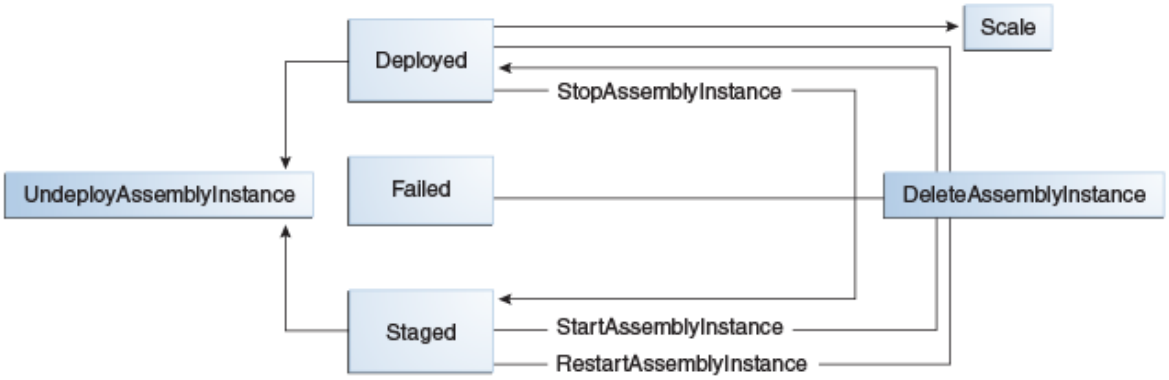
1.2.12 Deployment Life Cycle

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.

At deployment time, you choose the assembly instance to be deployed.

Deployment of an assembly instance will transition through various phases (Figure 1–4). The phases include: *Staged*, *Deployed*, and *Failed*. Each state allows a subset of operations. For example, when an assembly instance is deployed, you may start and stop the assembly, 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 1–4 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 (a scaling group corresponds to an appliance from the original assembly). 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.
  - *Suspend* This operation suspends an assembly instance.
  - *Resume* This operation resumes a suspended assembly instance.
- *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.

2 Architecture

This chapter describes the architecture of Oracle Virtual Assembly Builder, and contains the following sections:

- Section 2.1, "Major Components"
- Section 2.2, "Setup Scenarios"
- Section 2.3, "Deployment Platforms"

2.1 Major Components

This section describes the major components of Oracle Virtual Assembly Builder.

2.1.1 Oracle Virtual Assembly Builder Studio

Oracle Virtual Assembly Builder Studio is the component that allows users to create assemblies, and has two interfaces, the Oracle Virtual Assembly Builder Studio GUI and the `abctl` command-line interface.

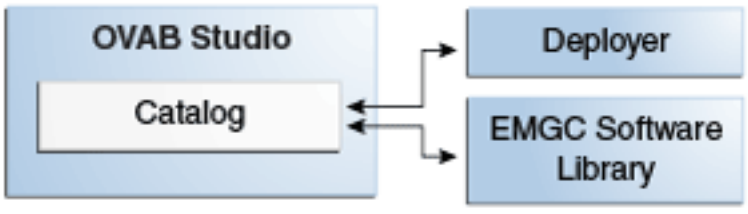
Oracle Virtual Assembly Builder Studio and `abctl` interact with Oracle Virtual Assembly Builder Deployer as a Web client as shown in Figure 2–2. Oracle Virtual Assembly Builder Studio provides a Deployer user interface whereby deployments can be initiated from Oracle Virtual Assembly Builder Studio and leverage the Oracle Virtual Assembly Builder Deployer.

2.1.1.1 High-Level Catalog Overview

The Oracle Virtual Assembly Builder Studio catalog contains the metadata definitions of appliances and assemblies that are the result of introspection. The captured file sets, virtual machine templates and deployment plans for those appliances and assemblies are also stored in the catalog.

Figure 2–1 shows Oracle Virtual Assembly Builder Studio and its relationship to both its catalog as well as to the Deployer and Oracle Software Library.

Figure 2–1 Oracle Virtual Assembly Builder Studio and Deployer



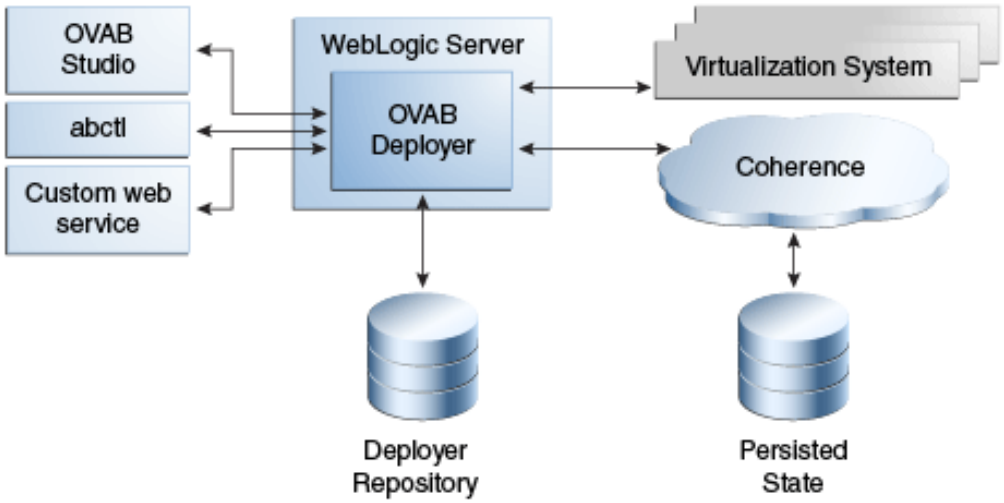
2.1.2 Deployer

The Oracle Virtual Assembly Builder Deployer is a Java EE application that maintains *assembly archives* created by Oracle Virtual Assembly Builder Studio, provides operations for registering assembly archives to virtualized systems such as Oracle VM and provides operations 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 Deployer runs in a servlet container in the WebLogic Server Administration Server. Figure 2–2 shows the top-level components of the Oracle Virtual Assembly Builder Deployer.

Figure 2–2 Deployer Architecture



2.1.2.1 Web Application

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

Operations against the Web service are made by posting an HTTP request to the Oracle Virtual Assembly Builder 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. The response is usually an XML document related to the operation performed, except in cases where artifacts such as assembly archives, deployment plans or metadata files are being downloaded from the Deployer to the client. See *Oracle Virtual Assembly Builder Developer's Guide* for sample requests and responses.

Some operations may define an asynchronous action.

2.2 Setup Scenarios

This section describes the different ways that you can set up Oracle Virtual Assembly Builder.

2.2.1 Deployer-only Installation Scenario

You can set up Oracle Virtual Assembly Builder Deployer by itself, without Oracle Virtual Assembly Builder Studio. This installation scenario provides you the ability to configure deployment targets, upload assembly archives to Deployer, create assembly instances, deploy/undeploy/start/stop assembly instances and scale appliance instances.

This configuration contains the following interfaces:

- You can use the Web service API to perform either PaaS or administrative operations on the Deployer.
- You can use abctl to interact with the Web service API for either PaaS or administrative operations on the Deployer.

2.2.2 Deployer Not Co-located with Studio

You can set up Oracle Virtual Assembly Builder Studio to interact remotely with Oracle Virtual Assembly Builder Deployer.

This installation scenario provides you the ability to create appliances and assemblies, create appliance templates and assembly archives and create deployment plans.

This configuration contains the following interfaces:

- The Oracle Virtual Assembly Builder Studio graphical user interface, and
- `abctl` CLI (all functionality for creating and deploying assemblies)

2.2.3 Deployer Co-located with Studio

You can set up Oracle Virtual Assembly Builder Deployer co-located with Oracle Virtual Assembly Builder Studio on the same machine, but in a different process. This configuration contains the previously described interfaces for Oracle Virtual Assembly Builder Deployer and Oracle Virtual Assembly Builder Studio, and provides you the ability to perform all the operations described in the "Deployer-only Installation Scenario", plus those operations in "Deployer Not Co-located with Studio".

2.3 Deployment Platforms

You can deploy Oracle Virtual Assembly Builder assembly archives to the Oracle VM 3.2 or higher platform.

3 Security

This chapter describes the security of Oracle Virtual Assembly Builder, and contains the following sections:

- Section 3.1, "Resources"
- Section 3.2, "Roles and Groups"
- Section 3.3, "Security Model Employed by Deployer"

3.1 Resources

The resources describe in Table 3–1 are protected:

Table 3–1 Resources

Resource	How protected
Target	For Oracle VM, a target is created by the CloudAdmin. For Oracle VM, the CloudAdmin grants permission to ApplicationAdmins to use a target. In Oracle VM, the CloudAdmin provides shared credentials for Oracle VM Manager. Only a CloudAdmin may view the configuration information of a target.
Credentials (passwords and keys)	Encrypted
Assembly archive in the Deployer	Protected by the "owner" concept, see Section 3.3.2, "Assembly Archive Authorization".
Assembly instances and deployment plans in the Deployer	Protected by the "owner" concept, see Section 3.3.2, "Assembly Archive Authorization".

3.2 Roles and Groups

Oracle Virtual Assembly Builder defines security roles and groups. The product installer sets up the roles and groups for the embedded LDAP case and you create the users and add them to the groups using the Oracle WebLogic Server Administration Console.

Follow this process to create roles and groups:

1. Use the procedures in *Oracle® Fusion Middleware Securing Oracle WebLogic Server* to configure Oracle WebLogic Server with an external LDAP server.

2. The groups "CloudAdmins" and "ApplicationAdmins" are automatically created at installation. Add the users defined in the LDAP server to these groups.
3. Place the groups into the security roles using the role expression `Grp(GroupName|GroupName|GroupName)`.

   The process of computing and granting roles in Oracle WebLogic Server is referred to as role mapping. An access decision is the operation made by an Authorization provider that determines whether a subject has permission to perform a given operation on a WebLogic resource. (See "Access Decisions" in *Developing Security Providers for Oracle WebLogic Server* for more information.

3.2.1 ApplicationAdmin Group

The `ApplicationAdmins` group is created automatically at installation.

You create `ApplicationAdmin` users out-of-band in LDAP. You include members of this group into the `ApplicationAdmins` group.

3.2.2 CloudAdmins Group

The `CloudAdmins` group is created automatically at installation.

You create `CloudAdmin` users out-of-band in LDAP. You include the users in the `CloudAdmins` group, which results in the `CloudAdmin` role being assigned at login.

3.2.3 CloudAdmin Role

You create the `CloudAdmin` role by an auth-constraint in the web.xml and a security-role-assignment of the `CloudAdmins` group to the `CloudAdmin` role in weblogic.xml.

3.2.4 ApplicationAdmin Role

You create the `ApplicationAdmin` role by an auth-constraint in the web.xml and a security-role-assignment of the `ApplicationAdmins` group to the `ApplicationAdmin` role in weblogic.xml.

3.3 Security Model Employed by Deployer

Oracle Virtual Assembly Builder Deployer is constructed as an application running in the Oracle WebLogic Server container and leverages the security infrastructure provided by Oracle WebLogic Server. Oracle WebLogic Server is configured with the embedded LDAP authenticator by default but you can reconfigure Oracle WebLogic Server to point to an external corporate LDAP. The Deployer depends upon Oracle WebLogic Server users being put into one or both of two groups: CloudAdmins or ApplicationAdmins.

For information on creating the CloudAdmin and ApplicationAdmin groups and roles, see Section 3.2, "Roles and Groups".

Having a user in those groups allows them to be mapped to having the roles needed by the Deployer: CloudAdmin or ApplicationAdmin. The Deployer's access control requires that a user be in one or both of those groups. When you set up a connection to the Deployer using `abctl` you must specify a username and password for an Oracle Weblogic Server user that has been added to one or both of those groups.

When a user attempts a Deployer operation using either Oracle Virtual Assembly Builder Studio or `abctl`, they are authenticated using the connection information, and

then once authenticated, their request goes to the Deployer's Servlet running in the WLS Servlet Container. The servlet checks that they are in one of those roles and then performs additional checks (for example, whether the user is allowed to access the specified target, or whether the user is allowed to access the specified assembly archive).

Oracle Virtual Assembly Builder Deployer uses Oracle WebLogic Server capabilities for access checking, however, Oracle Virtual Assembly Builder Deployer does not have visibility on the LDAP information for managing those identities. Due to this circumstance, it is possible for you to delete a user out of Oracle WebLogic Server while Oracle Virtual Assembly Builder configuration referencing that username is still in place.

Caution: Once an Oracle Virtual Assembly Builder request has moved beyond the authentication and access checking described above, it will continue to completion even if the user information is removed from the WebLogic Server authentication store.

3.3.1 Target Authorization

For Oracle VM, only the CloudAdmin can perform functions such as creating a target, or viewing the configuration information of a target. For Oracle VM, the CloudAdmin grants permission to ApplicationAdmins to use a target.

In Oracle VM, the CloudAdmin provides shared credentials for Oracle VM Manager.

Caution: If a user is removed from the Oracle WebLogic Server authentication store, you must remove any cached information about that user from the Deployer by removing that user from any targets (`describeUserTargets` and `removeTargetUsers`) to which they were previously added.

Removing them from the targets in the Deployer removes any cached information for that user from the Deployer. This prevents a situation where a user has cached information, the user is removed from Oracle WebLogic Server, a different user of the same name is added into Oracle WebLogic Server and the new user inherits all the previous user's cached information in the Deployer.

3.3.2 Assembly Archive Authorization

When you attempt to access an assembly archive, the Deployer performs a check. The following users are granted access:

- A user with the CloudAdmin role.
- The owner of the assembly archive. An assembly archive corresponds 1:1 with an assembly version and an assembly may have multiple assembly versions. The user who uploads the first assembly archive for an assembly (version 1.0) owns the assembly.
- Additional users can be granted access to an assembly.

3.3.2.1 Authorizing Additional Users with the addAssemblyUsers Command

You can authorize additional users, adding them to the access list, using the `addAssemblyUsers` command. Only a user with the CloudAdmin role or the assembly archive owner can perform this operation.

Only the ApplicationAdmin who is the owner of the assembly archive can manage it by adding/removing/describing assembly users, deleting the assembly or updating the assembly archive. The users in the assembly access list can use the assembly by viewing, downloading, registering and creating assembly instances (assuming they also have access to the target).

3.3.2.2 Assembly Instances

The user who creates an assembly instance is its owner. Besides the CloudAdmins, only the assembly instance owner can manage and use an assembly instance.

3.3.2.3 Assembly Resources

The `uploadAssemblyResources` command is controlled by a security policy. A resources file may or may not contain scripts. If the resource file does not contain scripts, a user on the assembly access list can run the command. If the resource file does contain scripts, only the CloudAdmin user is allowed to run the command, to prevent a malicious attack.

When including scripts in the resources files, the lifecycle names that are supported are: `pre-deploy`, `post-deploy`, `deployer-pre-app-config`, `deployer-post-app-config`, `deployer-pre-vm-start`, `deployer-post-vm-start`, `deployer-pre-vm-stop`, `deployer-post-vm-stop`, `pre-undeploy`, `post-undeploy`. You can create corresponding script folder names.

3.3.3 Enabling the Deployer's Authentication and Authorization Model

Perform these steps to enable the Deployer's authentication and authorization model:

1. A system administrator defines users in LDAP and assigns a `CloudAdmin` or `ApplicationAdmin` role to those users. These roles control what things a given user can do. This step is done through the Oracle WebLogic Server administrative console.

   You perform the remainder of the operations against the Web service; the Web service calls require Oracle Virtual Assembly Builder user credentials. The Deployer Web service operations that need to be called are as follows:

2. `createTarget` - This operation, which can only be performed by the CloudAdmin, defines the connection information, and, depending on the backend type, user credentials for the backend. For Oracle VM, credentials are supplied here.

3. `addTargetUser` - Depending on the backend type, this may be a CloudAdmin call or ApplicationAdmin call, due to differences in the security models of the backend systems.

   - For Oracle VM targets, this operation is only performed by the CloudAdmin and is used to control what users can access the pool. This is a CloudAdmin operation because the credentials supplied by the CloudAdmin must be protected from general users.

4. `uploadAssemblyArchive` - This call may be made by either a CloudAdmin or ApplicationAdmin user. The first user to upload the archive is the owner of the

archive. Only the archive owner or users added using the `addAssemblyUser` command may upload new versions of the archive or create assembly instances based on the archive.

For procedures for configuring Security refer to Section 5.4.2, "Configuring Targets".

4

Appliance and Assembly Structure

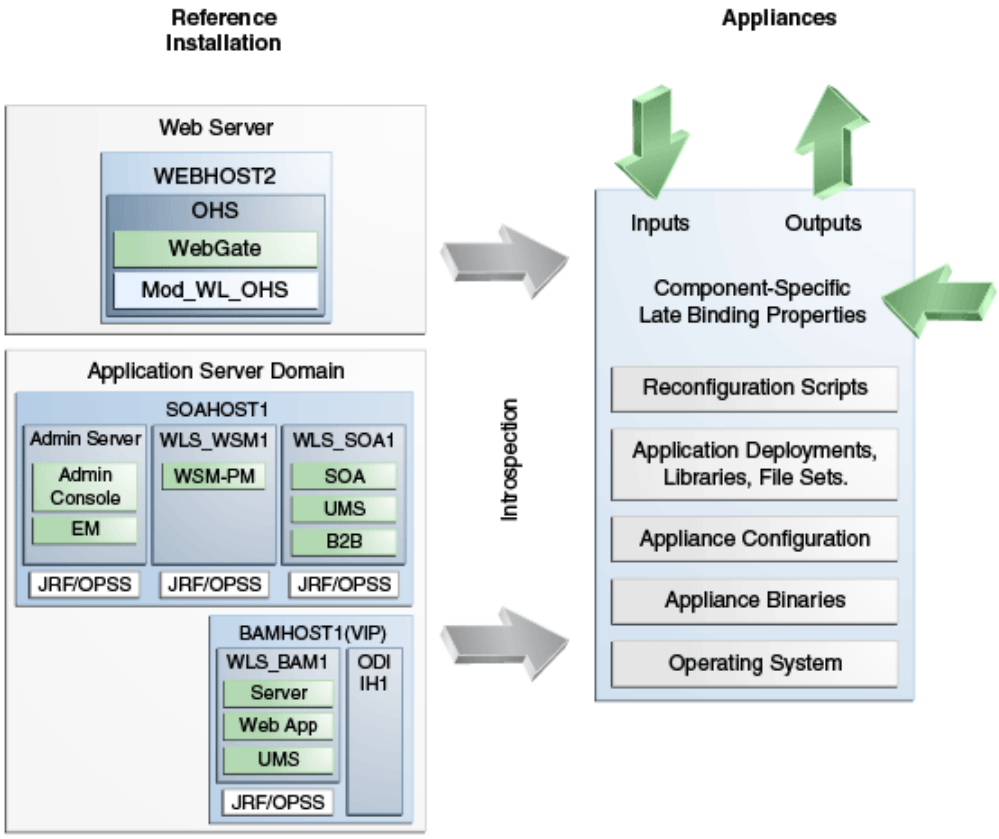
This chapter describes the structure of appliances and assemblies in Oracle Virtual Assembly Builder, and contains the following sections:

- Section 4.1, "Appliance Structure"
- Section 4.3, "New Structures for Deployments"

4.1 Appliance Structure

An appliance is a set of self-contained virtual disk images with all the software to run a single instance of a single component. An appliance includes appliance configuration, appliance binaries, and an operating system (Figure 4–1).

Figure 4–1 Appliance Structure



4.1.1 Appliance Configuration

Appliance configuration is captured during introspection of the appliance, and includes introspection properties, system and user properties, file sets, and optionally, customized scripts and properties files.

4.1.2 Appliance Binaries

Appliance binaries are software binaries included in the appliance. These include both the container or product binaries as well as the configured customer application binaries.

4.1.3 Operating System

A customer provides the operating system base image for the appliance, based on an Oracle Enterprise Linux template, and customized for his particular requirements. Oracle Virtual Assembly Builder has requirements for the base image which are documented in *Installing Oracle Virtual Assembly Builder*. Oracle Virtual Assembly Builder publishes several sample base images.

4.2 Assembly Structures

Appliances can be grouped, and their relationships defined into artifacts called software *assemblies* which provide a blueprint describing a complete multi-tier application topology.

Assemblies include appliances, as well as references to external systems which represent infrastructure such as databases, servers or security providers that cannot or should not be included in an assembly.

4.3 New Structures for Deployments

This section describes structures available for deployments:

- Section 4.3.2, "Configuring your Network Interface (NIC)"
- Section 4.3.3, "Shared File Sets"
- Section 4.3.4, "Zero-count Appliances"
- Section 4.3.5, "Anti-Affinity"

4.3.1 Terminology

The following alphabetical list of terms are defined:

- anti-affinity: placing multiple instances of a particular appliance across different physical hosts in a target, with even distribution of instances across physical machines.
- NIC: network interface. A NIC may only be connected to one Vnet.
- shared file sets: file sets shared with individual appliances within assemblies.
- Vnet: virtual network. One or more NICs or vNICs connect to the Vnet.
- vNIC: virtual network interface. A vNIC may only be connected to one Vnet.
- zero-count appliance: an appliance initially deployed with zero appliance instances.

4.3.2 Configuring your Network Interface (NIC)

Creating a new assembly automatically creates one Vnet for that assembly. That Vnet is the default virtual network for the assembly and represents an actual network in the deployment environment. You can create additional Vnets (if you have additional networks in your deployment environment), or delete ones that you have created. You can bind network interfaces on appliances to a Vnet during assembly editing.

4.3.3 Shared File Sets

If supported by the underlying infrastructure platform, you can specify that file sets be shared with individual appliances within assemblies.

4.3.4 Zero-count Appliances

You can initially deploy an appliance with zero appliance instances. However, in subsequent scaling operations you could add appliance instances to those appliances that are part of the assembly configuration but were initially "deployed" with a zero-instance count. Only appliances that no other appliance in the assembly references can have a zero-instance count. This means no other appliance depends on it or connects to any of its inputs.

4.3.5 Anti-Affinity

You can specify the requirement to place multiple instances of a particular appliance across different physical hosts in a target.

You can the maximum number of appliance instances that can be deployed on each physical machine. For example, an anti-affinity of 1 ensures that each instance of an appliance is deployed onto a different physical machine, while an anti-affinity of 2 means a maximum of two appliance instances can be deployed on each physical machine - thus six appliances require three physical servers.

There must be enough physical servers to distribute the appliances instances with a maximum of *n* appliance instances per physical server. An anti-affinity of 2 means six appliance instances require at least three physical servers (each server does not have to have two appliances - this is only the maximum). Eight appliance instances would require at least four physical machines.

The benefit of this feature is limiting your exposure to failure. If a physical server goes down, you do not lose the entire appliance and cause the assembly to go down.

5

Using Oracle Virtual Assembly Builder

This chapter describes how to use Oracle Virtual Assembly Builder, and includes the following sections:

5.1 Oracle Virtual Assembly Builder Interfaces

Oracle Virtual Assembly Builder provides the following user interfaces depending on which parts of the product you have installed:

- When you have installed Oracle Virtual Assembly Builder Studio the following interfaces are supported:
  - Oracle Virtual Assembly Builder Studio graphical user interface
  - `abctl`, a command-line tool.
  - Oracle Virtual Assembly Builder Studio and `abctl` interact with Oracle Virtual Assembly Builder Deployer as a Web client. In a suitably configured Oracle Virtual Assembly Builder Studio environment, you can perform deployment through the configured connection to Oracle Virtual Assembly Builder Deployer using either the graphical user interface or abctl.
- When you have installed Oracle Virtual Assembly Builder Deployer the following interfaces are supported:
  - `abctl`, a command-line tool.
  - A Web service API to interface with Oracle Virtual Assembly Builder Deployer Web service, described in *Oracle Virtual Assembly Developer's Guide*.

  When only Oracle Virtual Assembly Builder Deployer is installed, you can access only a subset of the commands through the command-line interface (all of the Web service operations are available, however). When Oracle Virtual Assembly Builder Studio is installed, or Oracle Virtual Assembly Builder Studio and Oracle Virtual Assembly Builder Deployer are installed, you can access all the commands. See Appendix A, "Command Line Reference".

> **Note:** You cannot launch two sessions of either the Oracle Virtual Assembly Builder Studio or `abctl` interfaces at the same time.

5.1.1 Accessing Oracle Virtual Assembly Builder Studio

Launch Oracle Virtual Assembly Builder Studio by executing the command:

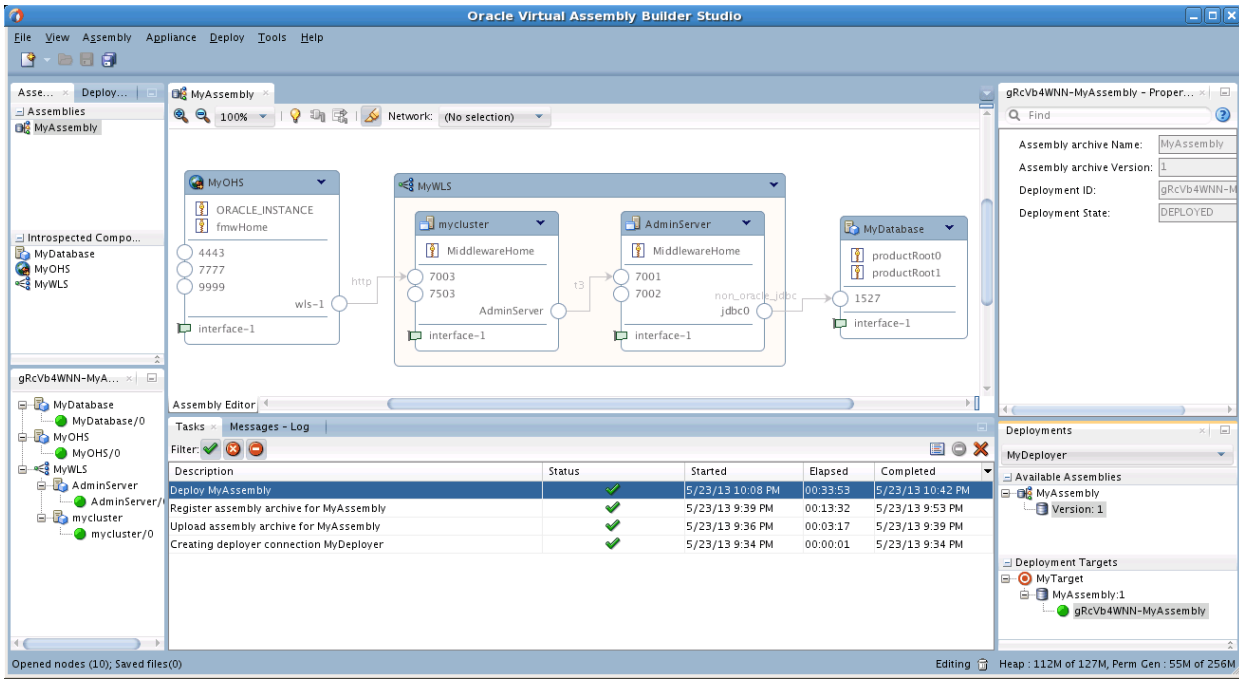
```
$AB_INSTANCE/bin/abstudio.sh
```

Where, `$AB_INSTANCE` is an instance of Oracle Virtual Assembly Builder Studio, in the format *ab_instance<instance number>*, installed in an Oracle Home.

Figure 5–1 shows Oracle Virtual Assembly Builder Studio.

There are multiple ways to perform operations in Oracle Virtual Assembly Builder Studio. You can use context menus by right clicking on a node, or you can use the main menu options. The Assembly main menu provides assembly related operations, the Appliance menu provides appliance related operations, and the Deploy menu provides Deployer related operations.

Figure 5–1 Oracle Virtual Assembly Builder Studio



Oracle Virtual Assembly Builder Studio

****

5.1.2 Accessing the abctl Command-Line Tool

Launch the `abctl` command-line tool by executing the command:

```
$AB_INSTANCE/bin/abctl
```

5.1.3 Accessing Logs

The log file is stored at `$AB_INSTANCE/logs/assemblybuilder.log`.

Access the log file manually, or view its messages through Oracle Virtual Assembly Builder Studio in the *Messages* window.

5.1.4 Shutting Down Oracle Virtual Assembly Builder Studio

When you close Oracle Virtual Assembly Builder Studio, if a long running process which interacts with the local catalog is in progress, a warning dialog indicates that operations are in progress. You can decide whether to continue or abort the shutdown. You are recommended to allow the operations to complete before shutdown, to avoid a problem on restart.

5.1.5 Differences Between the Interfaces

The Oracle Virtual Assembly Builder Studio and `abctl` interfaces complement each other but do not include identical functionality. Here are the main differences:

- When running in Deployer-only mode only the `abctl` interface is supported.
- Only Oracle Virtual Assembly Builder Studio provides editing capability. That is, the following operations are not supported in `abctl`:
  - managing file set definitions: updating file set definitions
  - creating/editing a deployment plan
- In Oracle Virtual Assembly Builder Studio, you can introspect multiple reference systems and put the results into a new or existing assembly. In `abctl`, you must introspect reference systems one-by-one and subsequently add them to an assembly.
- Only `abctl` provides the ability to create a 'target' connection to an Oracle VM 3.0 environment. Oracle Virtual Assembly Builder Studio does not have a connection wizard.

These differences are further detailed in Section 5.3, "Operations Related to Creating an Assembly".

5.1.6 Naming Rules

Any user-provided names must follow these rules:

- The name must begin with an alphabetic character.
- The name may only contain alphanumeric characters, or the underscore (_) or hyphen (-) characters.
- The name must be 4 to 40 characters long.

5.1.6.1 Naming Conflicts

You may experience a name conflict between appliances or assemblies in a catalog if you import an appliance or assembly into a catalog where you already have an appliance or assembly with the same name. Mixed-case names are allowed, although the Oracle Virtual Assembly Builder catalog is case-insensitive. For example, "myAssembly" and "myassembly" are considered to be the same.

If you want to overwrite the existing appliance or assembly you can use the `force` option.

5.1.7 Symbolic Links

Symlinks are not supported by Oracle Virtual Assembly Builder, and can lead to errors during introspection, capturing file sets, and deployment. Avoid symlinks in your Linux reference systems.

5.2 Typical Workflow

Users will typically use Oracle Virtual Assembly Builder in these ways:

- Create assemblies and appliances:
  - introspect a reference system to capture the necessary metadata and configuration information for all components that make up the appliances within an assembly.
- Edit assemblies and appliances to configure the relationships among the appliances and any external resources.
  - create networks within an assembly
  - create network interfaces within an appliance
  - bind appliance inputs to network interfaces and bind network interfaces to networks
  - create external resources from an appliance output
- Creating an archive for the assemblies:
  - Create bootable virtual machine disk images with customized Oracle Enterprise Linux operating system distributions and configurable metadata allowing for deploy-time customization of the software component
- Define connections to the backend virtualization system:
  - Define the connection to Oracle VM backend endpoints and add deployment targets in the backend.
- Deploy: deploy the assembly into your environment.
  - create and edit a deployment plan
  - upload assembly archive to Deployer repository
  - register an assembly archive to a target
  - deploy assembly instances
  - perform other lifecycle operations on assembly instances

5.3 Operations Related to Creating an Assembly

This section details how you will use Oracle Virtual Assembly Builder Studio or `abctl` command line utility to perform operations related to creating an assembly.

- Section 5.3.1.3, "Managing and Discovering Plug-ins"
- Section 5.3.1.4, "Preparing Custom Scripts"
- Section 5.3.1.5, "Preparing Properties"
- Section 5.3.2, "Creating an Assembly"
- Section 5.3.1, "Introspecting Appliances"
- Section 5.3.3, "Creating Templates for an Appliance or an Assembly"
- Section 5.3.10, "Creating an Assembly Archive"
- Section 5.3.4, "Editing an Assembly Using Oracle Virtual Assembly Builder Studio"
- Section 5.3.5, "Editing an Assembly Using abctl"
- Section 5.3.6, "Clearing Assembly Passwords"

- Section 5.3.7, "Copy an Appliance or Atomic Assembly"
- Section 5.3.8, "Export Operations"
- Section 5.3.9, "Rename an External Resource"

5.3.1 Introspecting Appliances

You can introspect one or more appliances into the top-level catalog and optionally include the appliances into an assembly. During introspection, the metadata for appliances is created in the `$AB_INSTANCE/catalog/metadata` directory. A unique ID (called the capture ID or *cid*) is generated for each appliance, and is stored in its metadata. In addition, a file set definition is created in the shared area of the catalog.

When you add a shared appliance to an assembly, the metadata is copied and subsequent modifications you make to the shared appliance's metadata (the one shown in the Introspected Components panel of the Assembly navigator) are not reflected in any assembly that has a copy of that same appliance. Similarly, any change you make to the metadata of a copy of a shared appliance do not affect the source instance.

While the metadata is not shared, the file sets and templates associated with the various copies of the appliance are shared between all copies.

Note: You should not change any configuration or content of the reference system between introspection and capturing file sets, as that may create undesired results. For instance, introspecting a reference system on one date and capturing file sets in the "same" reference system at some arbitrary future date is not supported.

The available appliances supported for introspection are the set of supported Oracle product plug-ins, or a *generic* product that you capture products generically by taking as input a set of product directories to capture and a set of scripts to run on the appliance instance. You can also capture information to create appliance properties, inputs, and outputs.

The following appliances are supported for introspection:

- Generic Product
- Oracle Coherence*Web (Alias of WLS)
- Oracle Forms Services (Alias of WLS)
- Oracle HTTP Server
- Oracle Real Application Clusters Database
- Oracle Reports (Alias of WLS)
- Oracle Single-Instance Database
- Oracle RAC Database
- Oracle SOA Platform Plugin (Alias of WLS)
- Oracle Traffic Director
- Oracle Tuxedo 11g
- Oracle WebLogic Server

For introspection to succeed, some introspection plug-ins have specific requirements for the reference system's running state. Table 5–1 lists the preconditions for the products supported by Oracle Virtual Assembly Builder.

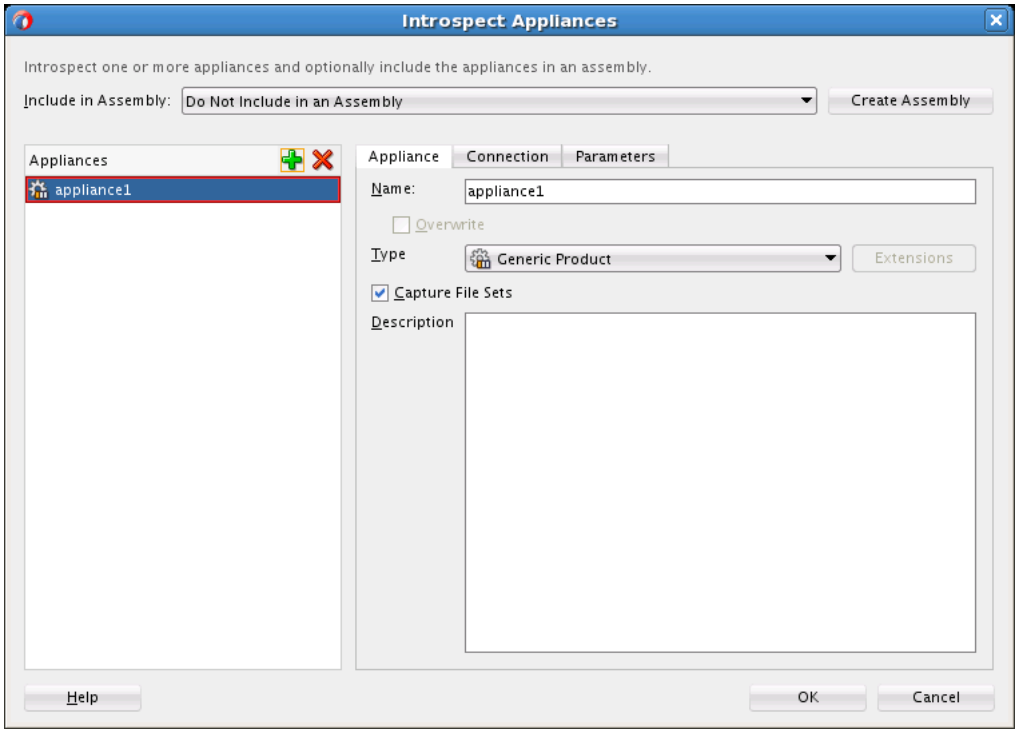
Table 5–1 Introspection Plug-in Requirements

Introspected Product	Running State Pre-Condition
Oracle WebLogic Server	Administration Server must be up and in the running state (not in the admin state). Managed Server(s) may be up or down.
Oracle Coherence*Web	Administration Server must be up and in the running state (not in the admin state). Managed Server(s) may be up or down.
Oracle Forms*Web	Administration Server must be up and in the running state (not in the admin state). Managed Server(s) may be up or down.
Oracle SOA for WebLogic Server	Administration Server must be up and in the running state (not in the admin state). Managed Server(s) may be up or down.
Oracle HTTP Server (OHS)	No requirement; Oracle HTTP Server may be up or down.
Oracle Single-Instance Database	In the introspection phase, the database can be up or down.
Oracle RAC Database	In the introspection phase, the database can be up or down.
Oracle Reports	Administration Server must be up and in the running state (not in the admin state). Managed Server(s) may be up or down.
Oracle Traffic Director	In the introspection phase, the Oracle Traffic Director application can be up or down.
Oracle Tuxedo	In the introspection phase, the Tuxedo application can be up or down.

5.3.1.1 Introspect Using Oracle Virtual Assembly Builder Studio

The Introspect Appliances wizard (Figure 5–2) is a standalone interface to allow you to introspect appliances into the top-level catalog and select whether to place them into an assembly. To access it, select **File** > **New** > **Appliance Introspection**.

Figure 5–2 Introspect Appliances Wizard



This graphic displays the Introspect Appliances window.

****

Enter the following information:

- Include in Assembly: Select whether to place the appliances into an assembly after a successful introspection. Choose a Parent Assembly from the drop-down list or select **Do Not Include in an Assembly**.

  If you choose to introspect an appliance into an assembly, the appliance becomes owned by the assembly and cannot be included in any other assembly. These appliances appear in the Introspected Components panel in the Assembly navigator.

  If you do not introspect into an assembly, you can add the resulting appliance to any number of assemblies. These appliances appear in the Introspected Components panel in the Assembly navigator.

  If the assembly does not exist, you can click **Create Assembly** to open the Create Assembly dialog and create a new assembly; you can then select the new assembly as the target assembly for the introspection.

- On the Appliance tab, configure:

  – Name: name your appliance. The name can be 4 to 40 characters, may not start with a digit, and no spaces or special characters are allowed (underscores are allowed). Assembly and appliance names are not allowed to be localized.

  – Overwrite: If introspecting at the top level, check this box to overwrite any top-level assembly or appliance object, provided that it is not registered.

    If you are introspecting into an existing assembly, checking this box overwrites only assemblies and appliances inside that assembly.

  – Type: select the type of appliance from the list.

  - Capture File Sets: check the box to capture file sets during introspection. You can capture file sets during either introspection or when creating a template. See Section 5.3.3.4, "Capturing File Sets".
  - Description: enter an optional description.

- On the Connection tab tab, configure:
  - Name: enter the name of the host that you want to introspect.
  - Host Name: enter the hostname of the host that you want to introspect. You can use Local Host to introspect locally.
  - Port: enter the port number for connecting to this host. The default port number is 22.
  - Username: enter the username for the SSH user to log into the remote host. This user must have permissions to access the introspected configuration.
  - Authentication: select **Password** and enter a password or select **Private Key** to reference the SSH key to use rather than providing a password. If selecting Private Key, select the browse button and navigate to the location of a private SSH key file on the local machine. The use of a private key file provides added security because no password handling is required by Oracle Virtual Assembly Builder.
  - Run As User: enter a name of a user on the remote machine to sudo as before executing operations. For example, if you log in with 'User Name' bob and 'Run As User' jill, the introspection process will run as jill, not bob. In that case, bob must do a sudo operation to jill.
  - Working Directory: enter the path to a directory on the remote host in which Oracle Virtual Assembly Builder may stage files required for introspection. The files may be reused.
  - Cleanup Working Directory: check to remove the artifacts copied over to the Working Directory once the introspection is complete.
  - Test Connection: click the **Test Connection** button to test if you can successfully connect to the host.

    Note: You cannot perform remote introspection of a database if you cannot log into the database machine with the database installation owner's account. If remote introspection is required, you must enable the account for remote access.

- On the Parameters tab, configure the required values for appliance parameters. Depending on the type of appliance chosen, different sets of properties are displayed. Set the properties for that appliance by selecting the cell for the property and entering a value for the property. Required properties are identified with an asterisk. See Appendix B, "Oracle Virtual Assembly Builder Introspection Plug-ins", for information on introspection properties.

5.3.1.1.1 Summary of Appliances for Introspection

Click **Finish** to begin the introspection.

You can see the progress of the introspection in the Tasks navigator. Oracle Virtual Assembly Builder Studio displays a node for the appliance being introspected. If introspection fails, a red X mark is displayed in the navigator. You can open the task log for the failed task from the Tasks navigator using the **View Task Log** toolbar button. See also Appendix D, "Troubleshooting".

5.3.1.2 Introspect Using abctl

`abctl` provides both local and remote introspection capability. For remote introspection, the Oracle Virtual Assembly Builder host must have SSH access to the subject machine.

The `-name` flag is optional.

Here are two examples:

Example 5–1 Introspect Oracle HTTP Server Remotely

```
$ ./abctl introspectOHS -oracleInstance /path/to/oi -componentName ohs1 -name
myOHS -remoteHost myReferenceSystemHost -remoteUser abdemo
```

Example 5–2 Introspect Oracle WebLogic Server Locally

```
$ ./abctl introspectWLS -wlsHome /path/to/wls/wlserver -domainRoot /path/to/user_
projects/domains/basic_domain -adminUser weblogic -name myWLS
```

For more information see Appendix A, "Command Line Reference", which contains the details of the parameters that can be passed into the command.

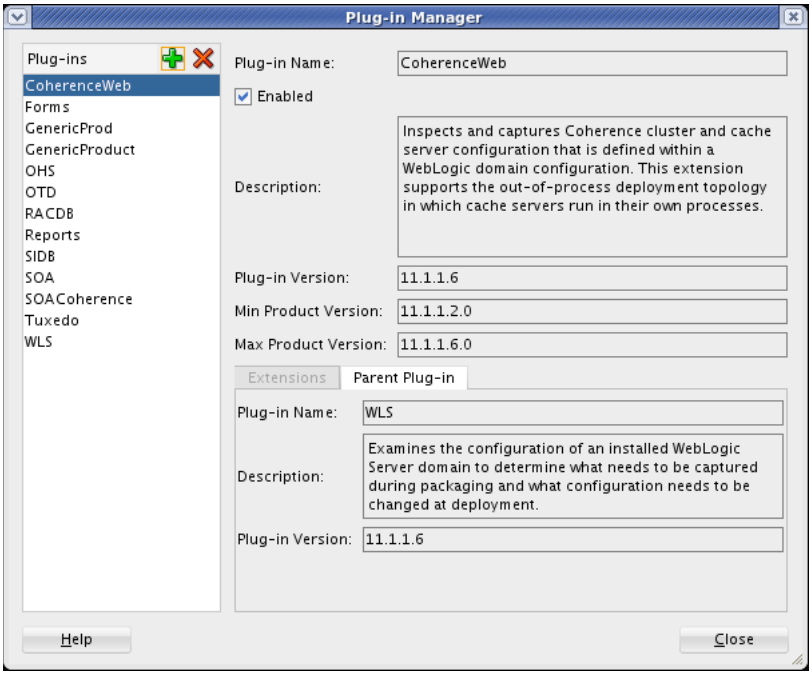
5.3.1.3 Managing and Discovering Plug-ins

Starting with Oracle Fusion Middleware 12c (12.1.2), each product's plugin for Oracle Virtual Assembly Builder is shipped with the respective product's release and laid out in that product's top level installation directory (ORACLE_HOME for Oracle products) at installation time. To allow a plug-in to be used by Oracle Virtual Assembly Builder, you must install the plug-in into the Oracle Virtual Assembly Builder environment. You can add a product plug-in to Oracle Virtual Assembly Builder at any time after installing Oracle Virtual Assembly Builder.

Oracle Virtual Assembly Builder provides a Plug-in Manager to allow you to manage introspection plug-ins, and download and install new Oracle Virtual Assembly Builder introspection plug-ins from a local or remote Oracle Home.

5.3.1.3.1 Managing and Discovering Plug-ins with Oracle Virtual Assembly Builder Studio The Plug-in Manager dialog allows you to view and manage all the plug-ins in the Oracle Virtual Assembly Builder installation. You can launch a discovery of a new plug-in by clicking the + button on the toolbar, enable or disable plugins by clicking the checkbox for a selected plug-in, or remove a selected plug-in by clicking on the X button. Disabling a base plug-in disables all of its extensions. You can access this dialog by selecting **Tools** > **Introspector Plug-in Manager**.

Figure 5–3 Plug-in Manager



This graphic displays the Plug-in Manager, which is described in the surrounding text.

****

To install a plug-in or extensions from a product home:

1. Launch a discovery of a new plug-in by clicking the **+** button on the toolbar.
2. Click **Next** in the welcome page of the Discover Plugins wizard.
3. Specify the connection information for the Oracle Home containing the plug-in:
   - Remote Host or Local Host:
   - If you selected Remote Host, configure the following information:
     - Host Name: Enter the name of the host that you want to introspect.
     - Port: Enter the port number for SSH for this host. The default port number is 22.
     - Username: Enter the username for the SSH user to log into the remote host. This user must have permissions to access the plug-in.
     - Authentication: Select **Password** and enter a password or select **Private** Key to reference the SSH key to use rather than providing a password. If selecting Private Key, select the browse button and navigate to the location of a private SSH key file on the local machine. The use of a private key file provides added security because no password handling is required by Oracle Virtual Assembly Builder.
     - Remote Working Directory: Enter the path to a directory on the remote host in which Oracle Virtual Assembly Builder may stage files required for plug-in discovery. The files may be reused.
     - Remote Cleanup: Check to remove the artifacts copied over to the Remote Working Directory once the plug-in discovery is complete.
4. Click **Test Connection** to verfiy you can successfully connect to the host.

5. Click **Next.**
6. Enter the Oracle Home or product root directory containing the plug-in(s).
7. Select one or more of the plug-ins to install into your base Oracle Virtual Assembly Builder installation. You can track installation progress through the Task Manager tab.

> **Note:** Some plug-ins may have a dependency on other plug-ins. If the installation of a particular plug-in fails, view the failure in the Task Manager, and reattempt the installation of the failed plug-in once the dependency is resolved.

8. Click **Finish**.

   You can verify the installation of the plug-ins from the Plug-in Manager.

5.3.1.3.2 Managing and Discovering Plug-ins with abctl Use the `describePlugins` command to list the set of installed introspector plug-ins and extensions including their status. Example 5–3 shows the `describePlugins` command:

Example 5–3 describePlugins Command

```
$ ./abctl help -command describePlugins
$ ./abctl describePlugins
```

Use the `enablePlugin` command to enable the specified introspector plug-in or extension. Example 5–4 shows the `enablePlugin` command:

Example 5–4 enablePlugin Command

```
$ ./abctl help -command enablePlugin
$ ./abctl enablePlugin -pn WLS -recurse
```

Use the `disablePlugin` command to disable the specified introspector plug-in or extension. Example 5–5 shows the `disablePlugin` command:

Example 5–5 disablePlugin Command

```
$ ./abctl help -command disablePlugin
$ ./abctl disablePlugin -pn WLS
```

Use the `installPlugins` command to install one or more plug-ins and extensions from a product home. You may be prompted to enter the password for the remote SSH user when you execute the command. Example 5–6 shows the `installPlugins` command:

Example 5–6 installPlugins Command

```
$ ./abctl help -command installPlugins
$ ./abctl installPlugins -productRoot myPath
```

For more information see Appendix A, "Command Line Reference".

5.3.1.4 Preparing Custom Scripts

Oracle Virtual Assembly Builder supports a common scripting API to provide custom scripts for the following features:

- *Custom reconfiguration scripts* - you can use the custom reconfiguration scripts feature to add scripts to an appliance that gets created by an existing introspector. For example, "When I capture my WLS installation, I want to add scripts to do additional work beyond what is already done on the VM where WLS is to be deployed."

  At introspection time, the scripts and properties are captured and added to the appliance. The captured scripts are executed alongside the creating plugin and extension on the vServer during the various reconfiguration operations.

- *Generic appliance (GenericProd) introspector plug-in scripts* - when you do not want to (or cannot) use an existing introspector plugin to capture and deploy a particular product, you can use the GenericProd introspector plugin to simply create an appliance that captures a set of product directories and a set of scripts. The captured product directories are present on the VM at the same location and the scripts are executed to reconfigure the product and start/stop. For example: "No introspector plugin exists for the product I want to capture, but it should be easy to write a few scripts to change the product's configuration to work in different environments.

You can supply one or more scripts to be executed during each of the various reconfiguration operations. The following reconfiguration operations are supported:

- Configuration - Executed during the initial vServer boot. This is where a product will be reconfigured to be usable on the newly deployed vServer.
- Start - Executed after the Configuration operation has completed at initial vServer boot and during all subsequent boots.
- Stop - Executed during shutdown of the vServer. This is where the product should be shutdown gracefully.
- Ping - Executed on demand by external applications to query the status of the deployed product running on the vServer. Only supported by generic appliance.

5.3.1.4.1 Script Root Directory The script root directory is a top level directory supplied by the user containing zero or more of the following sub-directories:

- A set of sub-directories containing scripts to be executed on the vServer.
- A directory named `properties/` that contains one or more property files.
- A directory named `endpoints/` that contains one or more endpoint files (only partly supported by custom reconfiguration scripts).

The script root directory need not contain all well-known sub-directories and well-known sub-directories that do exist may be empty.

The names of the script sub-directories vary depending on which scripting feature is being used. How this script root directory is supplied by the user also depends on which scripting feature is being used. See the respective documentation for details.

5.3.1.4.2 Script Sub-directories for Custom Reconfiguration Scripts Like `/etc/rc.d/`, each subdirectory will contain a set of one or more scripts which get executed at the appropriate time. The custom script sub-directories have the following well-known names:

- `vm-pre-app-config.d/` - executed only at first VM boot (initial deployment)
- `vm-post-app-config.d/` - executed only at first VM boot (initial deployment)
- `vm-pre-app-start.d/` - executed for all VM boots, follows `vm-*-app-config.d/` at first VM boot

- `vm-post-app-start.d/` - executed for all VM boots, follows `vm-*-app-config.d/` at first VM boot
- `vm-pre-app-stop.d/` - executed for all VM shutdowns.
- `vm-post-app-stop.d/` - executed for all VM shutdowns.

You can optionally create the `/ovab/scripts.d/` root directory and populate it with the set of well-known sub-directories and the scripts you want to execute. At the end of every introspection we check for the existence of this directory on the reference system and (if found) add these scripts to the appliance. Here is an example of a root custom script directory you can create:

```
/ovab/scripts.d/
    vm-pre-app-config.d/
        00configthis.sh
        01configthat.sh
    vm-post-app-config.d/
        00configotherthing.sh
    vm-pre-app-start.d/
        00startthisfirst.sh
        01startthatsecond.sh
    vm-post-app-start.d/
        00startotherthinglast.sh
    ...
```

The script sub-directories must only contain scripts that can be executed. The presence of a directory within a well-known sub-directory generates an error during introspection and an appliance is not created.

5.3.1.4.3 Script Sub-directories for Generic Appliance Scripts Each sub-directory may contain a set of one or more scripts which get executed at the appropriate time. The script sub-directories must have the following well-known names:

- `vm-app-config.d/` - executed only at first VM boot (initial deployment)
- `vm-app-start.d/` - executed for all VM boots, follows `vm-app-config.d/` at first VM boot
- `vm-app-stop.d/` - executed for all VM shutdowns
- `vm-app-ping.d/` - executed on demand from external applications

Here is an example of a root script directory that you can create:

```
/my/root/scriptdir/
    vm-app-config.d/
        00configthis.sh
        01configthat.sh
    vm-app-start.d/
        00startthisfirst.sh
        01startthatsecond.sh
    vm-app-stop.d/
        00stopthis.sh
    ...
```

The script sub-directories must only contain scripts that can be executed. The presence of a directory within a well-known sub-directory generates an error during introspection and an appliance is not created.

5.3.1.4.4 Script Execution Scripts under each sub-directory are captured during introspection and stored with the appliance. During reconfiguration operations the

appropriate set of scripts according to the requested operation are executed sequentially, in sorted order.

All scripts are executed as the root user to provide the flexibility of performing operations requiring root privileges or switching to another user as necessary.

All scripts must exit with a zero exit status upon success. Any script exiting with a non-zero exit status results in the failure of the operation. The operation blocks as each script executes and waits for the script to exit before continuing. The script must complete execution and exit in a timely manner.

The standard out (stdout) and standard error (stderr) output from each executed script is redirected and appended to a script-specific console log file on the VM within the `$AB_INSTANCE/logs` directory. The name of the console log is generated from a combination of the script name and the script sub-directory name as follows: `<script-subdir-name>_<script-name>.log`

5.3.1.4.5 Script Environment The following environment variables are set in the execution environment of all scripts:

- `$AB_PROPFILE_DIR`: The directory containing the property files added to the appliance and modified according to the metadata in the reconfiguration environment.
- `$AB_ENDPOINT_DIR`: The directory containing the endpoint files generated from the metadata in the reconfiguration environment.
- `$AB_CLI`: The full path to the Oracle Virtual Assembly Builder command line utility available to scripts executed on the vServer.

The current working directory in the environment is undefined and should not be relied upon during script execution.

The presence of other environment variables depends on which scripting feature is being used. See the respective documentation for details.

5.3.1.4.6 Property Directory The script root directory may contain a properties/ sub-directory. If a properties/ directory exists then all files in the directory will be captured and added to the appliance.

Files with a <filename>.userprops or <filename>.pwprops naming scheme are parsed as a property file. The <filename> part must not contain ':' as this character will serve as a delimiter in property name generation. Files within the properties directory that do not conform to the above naming schemes are blindly copied without parsing.

When the property files are obtained from the 'properties/' directory the property names are modified as they are added to the appliance as follows:

`<prefix>:<filename>:<propertyname>`

During deployment the properties are written back out to the property file(s) using their original names, including the values as edited within the appliance or overridden in the deployment plan. The properties files may be accessed by scripts during execution using the `$AB_PROPFILE_DIR` environment variable.

5.3.1.4.7 Property Files A property file must consist of zero or more lines where each line must be a property declaration, a comment, or a blank line. Property declarations take the form of <name>=<value> and must be contained on a single line. Ending a line with '\' will not result in line continuation. Property names may contain only letters, digits, '_' (underscores), '.' (periods), and '-' (dashes).

Each property in a property file is added as a user property into the appliance. Like any other user property, you can edit these properties in the generated appliance and override them in deployment plans.

All properties in '.userprop' files are added as type "STRING". All properties in '.pwprop' files are added as type "PASSWORD" and supplied values must be the encrypted form. See Section 5.3.1.4.10, "Encryption Support" for instructions on how to obtain the encrypted form of a value.

All properties are marked as 'required' in the appliance metadata. Property declarations without any assigned value (nothing after '=') are set to null in the appliance metadata, requiring that the user assign a value to that property prior to deployment.

5.3.1.4.8 Endpoint Directory The script root directory may contain an endpoints/ sub-directory. This sub-directory may contain files describing an ApplianceInput and ApplianceOutput to be added to the appliance as part of introspection. An ApplianceInput represents a product's configuration related to the the host and port on which the product listens and accepts connections. An ApplianceOutput represents a product's configuration related to the host and port on which the product initiates connections to other products (or to instances of itself on other machines).

Each endpoint is described in a separate file with the following naming strategy:

- For ApplianceInput: `<input-name>.input`
- For ApplianceOutput: `<output-name>.output`

> **Note:** Only GenericProd plugin can supply an endpoints directory during introspection for creations of endpoints. Both features have access to the endpoints directory created on the vServer for use during reconfiguration operations.

During reconfiguration, endpoint files are created for all ApplianceInput and ApplianceOutput of the appliance. In addition, files are created for all ApplianceInput of other appliances to which the appliance is connected.

5.3.1.4.9 Endpoint Files t The custom reconfiguration scripts feature cannot create endpoint files. The feature can only create read endpoint files created by the underlying plugin that created the appliance. By contrast, generic product can both create endpoint files at introspection, and read endpoint files (during reconfiguration operations).

The `*.input/*.output` files are comprised of properties expressed as key/value pairs, one per line, with the form of <key>=<value>. The keys are not guaranteed to be valid shell variable names so endpoint files cannot be sourced to create shell variables.

The following properties can be contained in *.input files:

```
- port : required
- protocols : required, comma separated list with no whitespace
- host : derived internally, available at rehydration userprop.
- userprop.<property-name> : optional, always type='STRING'
- pwprop.<property-name> : optional, always type='STRING'
- sysprop.<property-name> : optional, always type='STRING'
- original-port : optional
- original-host : optional
```

Properties denoted as 'required' must be specified in the files supplied at introspection. The 'host' property should not be specified by the user and is instead derived automatically during rehydration. Rehydration scripts should use the value of the 'host' property to set the listening address when configuring an ApplianceInput and to set the connection address when configuring an ApplianceOutput.

The following properties can be contained in *.output files:

```
- protocol : required
- connected-input : derived internally, available at rehydration
- userprop.<property-name> : optional, always type='STRING'
- pwprop.<property-name> : optional, always type='PASSWORD'
- sysprop.<property-name> : optional, always type='STRING'
- original-port : optional
- original-host : optional
- conn-userprop.<property-name> : optional
- conn-pwprop.<property-name> : optional
- conn-sysprop.<property-name> : optional
```

Properties denoted as 'required' must be specified in the files supplied at introspection.

All 'userprop.*' properties are endpoint user properties.

All 'sysprop.*' properties are endpoint system properties.

All 'pwprop.*' properties are endpoint user properties of type PASSWORD and must be supplied in encrypted form and will be returned in encrypted form. See Section 5.3.1.4.10, "Encryption Support" for instructions on how to obtain the encrypted form of a value.

The properties with the 'conn-' prefix are connection properties and they are for specifying requirements on connected inputs. An input is only compatible with a given output if it supports the protocol of the output and has the user and system properties that the output names in its connection properties. This is enforced when creating connections in appliance metadata. Omitting this information may permit the output to be associated with an incompatible input.

The original-host and original-port properties are useful in determining which inputs were connected to which outputs in the case where the inputs/outputs of appliances captured separately were originally connected to each other and need to be reconnected in appliance metadata.

The 'connected-input' property should not be specified by the user and is instead derived automatically during rehydration. Rehydration scripts should use the value of the 'connected-input' property to derive the name of the *.input file by appending '.input' to the name. The filename indicated by the connected-input property will take the form of:

```
<assembly-path>.<input-name>
```

where each component of <assembly-path> is separated by a '.' (period).

This input file just like any other *.input file but it is for another appliance and therefore prefixed with the assembly-path to avoid input name collision. For example, take an appliance at path 'mySite/otherAppliance' which has input 'otherInput' which is connected to an output named 'output1' on a target GenericProd appliance. The connected output file, 'output1.output' would contain the following property:

```
connected-input=mySite.otherAppliance.otherInput
```

The *.input file representing 'otherInput' would be located at the following full path (alongside all other *.input/*.output files):

```
$AB_ENDPOINT_DIR/mySite.otherAppliance.otherInput.input
```

5.3.1.4.10 Encryption Support To prevent the storage of cleartext passwords on disk all password properties must be supplied in encrypted form. During rehydration the password properties (which may have been modified during assembly editing or by deployment plan) are written back out to the property files also in encrypted form. OVAB provides an encryption utility for use during property file creation and a decryption utility for scripts to invoke during rehydration operations.

You can use the `encryptProperties` command of the abctl interface to encrypt password values during property file creation:

```
abctl encryptProperties -propertyNames String ... [-outputFile String]
```

Example execution with three properties, applied to pre-existing file with replacement of myprop3 property:

```
% abctl encryptProperties -propertyNames myprop1 myprop2 myprop3 -outputFile
allmysecrets.pwprops
Enter value for property 'myprop1':
Confirm entered value for property 'myprop1':
Enter value for property 'myprop2':
Confirm entered value for property 'myprop2':
Enter value for property 'myprop3':
Confirm entered value for property 'myprop3':

Changes applied to pre-existing file 'allmysecrets.pwprops'.
Added property 'myprop1'.
Added property 'myprop2'.
Replaced property 'myprop3'.
```

During script execution the 'decryptProperty' command of the utility indicated by the $AB_CLI environment variable may be used to decrypt property values:

```
$AB_CLI decryptProperty -encryptedValue String
```

Example execution from a script:

```
# encrypted value already obtained from property file

# execute decryption utility and store value in variable
THE_PASSWORD=$($AB_CLI -encryptedValue $THE_PASSWORD_ENCRYPTED)

# now ready to use $THE_PASSWORD to reconfigure a product
```

Use the standard help facility, `abctl help <command>` in the source environment to learn about the `encryptProperties` command. Use the `$AB_CLI help <command>` on the vServer to to learn about the `decryptProperty` command.

5.3.1.4.11 Additional Property Support Other components, especially in the case of custom reconfiguration scripts, may create properties in the appliance that need to be fetched during script execution. Since these values were not originally supplied from a property file they are not present in any of the property files during reconfiguration. You can fetch these properties directly from the appliance metadata. Your scripts can use the `getProperty` command of the $AB_CLI utility during rehydration operations:

```
$AB_CLI getProperty -propertyName String [-parentAssembly] [-systemProperty]
[-instanceId String]
```

The Deployer delivers to the VM certain configuration parameters that influence the behavior of reconfiguration operations. These parameters are separate from the appliance metadata but may be useful in rare cases to scripts. The `getConfigParam` command of the $AB_CLI utility may be used by scripts during reconfiguration operations:

```
$AB_CLI getConfigParam -paramName String [-instanceId String]
```

5.3.1.5 Preparing Properties

This feature allows an introspection to pick up properties from one or more property file on the reference system during introspection, and for those properties to be added to the resulting appliance/assembly as user properties so they can be edited.

During deployment, the set of original properties including any modifications by the user are made available to scripts provided by the user to perform custom processing based on those properties.

5.3.1.5.1 Properties Directory The properties directory is supported only for the custom reconfiguration scripts feature, and not for the generic appliance introspection scripts.

To get property files picked up automatically during dehydration, you can place the property files in the well-known directory:

```
/ovab/scripts.d/properties/
```

This directory must reside on the same machine as the underlying product that is being captured. Within that directory, property files must conform to the following naming scheme:

```
<filename>.userprops
```

The `<filename>` must not contain ':' as this character will serve as a delimiter in property name generation.

Files within the properties directory that do not conform to the above naming scheme will be blindly transferred with the Appliance without reading it to generate additional user properties. This allows the user to provide additional files that may contain internal properties or other information to aid in the processing of user properties during reconfiguration.

5.3.1.5.2 Property File A property file must consist of zero or more lines where each line must be a property declaration, a comment, or a blank line. More formally, a property file must comply with the following syntax:

```
property-file   = *line
line            = prop-decl | comment | blank-line
prop-decl       = name "=" value NL
comment         = *WS "#" *CHAR NL
blank-line      = *WS NL
name            = <any character in "a".."z", "A".."Z", "0".."9", "_", "-", ".">
value           = *XATTRCHAR
XATTRCHAR       = <any CHAR, escaped as necessary for XML element attribute
interpretation>
CHAR            = <any character, excluding CTL (and NL), but including WS>
CTL             = <any control character (octets 0 -31) and DEL (127)>
NL              = <platform dependent line termination sequence>
WS              = <white space character>
```

Any property file that does not comply with the above syntax rules will result in an error and an appliance will not be created.

Property declarations must be contained on a single line. Ending a line with "\" will not result in line continuation.

All properties will be marked as "required" in the appliance metadata. Property declarations without any assigned value (nothing after "=") will be set to null in the appliance metadata requiring that the user assign a value to that property prior to deployment.

Whitespace is not permitted anywhere to the left of "=" in a property declaration. Whitespace to the right of "=" is assumed to be part of the intended value and will be preserved.

Comments and blank lines are preserved at dehydration and will be reproduced when the file is regenerated at rehydration.

5.3.1.5.3 Property Names Each *.userprops file in the properties directory will be read and an appliance user property will be generated for each property in each file. Property names will be modified by adding a prefix to designate that the property is a custom property and that the property belongs to a specific properties file as follows:

- For custom reconfiguration scripts:

  `custom:<filename>:<propname>`

- For generic product appliances:

  `generic:<filename>:<propname>`

The `<filename>` part comes from the properties filename with the ".userprops" suffix removed. The `<propname>` part is copied directly without modification from the property name found in the property file.

The user, during editing, will see the entire property name. At reconfiguraiton when the property files are recreated, the "`custom:<filename>:`" prefix will be removed and will not appear in the property files (that is, the property names originally found in the files will be preserved in the recreated files).

5.3.1.5.4 Property Values As indicated in the Property File section above, property values must conform to the requirements of XML element attributes. Any necessary escaping of characters in property values is the responsibility of the user when creating property files.

5.3.1.5.5 At Deployment During reconfiguration, the user properties in the appliance will be traversed and all properties with a "`custom:<filename>:`" prefix will be added to a properties file under the indicated filename (with ".userprops" suffix added).

The order of properties and the comments within the original properties file are preserved in the regenerated properties file.

All generated properties files will be placed into the same directory. The full path to that directory will be passed to all reconfiguration scripts as an environment variable with the following name:

```
$AB_PROPFILE_DIR
```

5.3.2 Creating an Assembly

You can create a new empty assembly by selecting **File** > **New Assembly**. You can can add existing appliances to the assembly using the assembly editor, or introspect appliances into an assembly as described in Section 5.3.1, "Introspecting Appliances".

Configure the following fields for the assembly:

- Name: enter a name for the new assembly in the Assembly Name field.
- Overwrite: if an appliance or assembly with the same name already exists, and it has not been registered, you may overwrite it by checking the Overwrite checkbox.
- Description: optionally, enter a text description.
- Default Vnet Name: enter a name for a default Vnet (virtual network) to be created for this assembly. You can connect one or more NICs or vNICs to the Vnet.

Click **OK** to create the assembly.

5.3.3 Creating Templates for an Appliance or an Assembly

Template creation generates virtual machine templates that are ready to be deployed into virtualized platforms. In Oracle Virtual Assembly Builder, Oracle VM is the only supported platform. Oracle Virtual Assembly Builder supports Oracle Enterprise Linux guest OS for all appliances.

To create a template, you must provide a system base image that contains the operating system. You may create your own system base image if the sample system base image does not meet your needs.

If a product that is introspected contains files encoded with a specific character encoding, ensure that the system base image you use to create templates for the resulting appliance(s) contains the needed character encodings.

Oracle Virtual Assembly Builder provides a sample system base image for Oracle Enterprise Linux templates. When creating Oracle Enterprise Linux templates, Oracle Virtual Assembly Builder transparently invokes Oracle VM's `modifyjeos` tool to create the virtual machine templates. The tool allows you to modify or customize the base image (for example, adding disk space to the base image, or specifying certain RPMs). Refer to "System Base Images" in *Oracle Virtual Assembly Builder Installation Guide* for details on how to create a custom system base image.

5.3.3.1 Base Image Validation

Introspector plug-ins can express prerequisites for base images to be enforced at template creation time. Plug-ins can specify the behavior on a failed check - either an error (which causes the template creation to fail), or a warning (which is presented to you but allows the template creation to proceed).

5.3.3.2 Storage of Templates

Templates are stored in the Oracle Virtual Assembly Builder instance's catalog directory. Template creation must be done on an Oracle Virtual Assembly Builder Host, where Oracle VM's modifyjeos is installed.

Note: Base images are stored in either $AB_INSTANCE, or in $ORACLE_HOME. Here is the order of precedence for base image detection:

- location specified by `-baseImage` flag
- $AB_INSTANCE/templates/baseImage/OVM/OEL
- $ORACLE_HOME/templates/baseImage/OVM/OEL

5.3.3.3 Creating Templates Using Oracle Virtual Assembly Builder Studio

This operation allows you to create templates for an appliance. Select **View** > **Appliances** to view the available appliances. Right click an appliance and select **New** > **OVM Appliance Template**.

In the *Configure Base Image Options* page, choose the base image that serves as the default for the templates. Configure the following fields:

- Assembly Level Base Image Path: Specify a location for the base images by selecting the platform default, or by choosing a custom base image.
- Use the Platform Default: select if you want to use the default base image.
  - When configuring the OEL base image for Oracle VM the default location is: `$AB_INSTANCE/templates/baseImage/OVM/OEL`.
- Choose a Custom Base Image: select if configuring a custom base image. Click the browse icon, navigate, and select the base image.
- Include Root Password in Template: check to include the OS Root password in the template, and enter the password.

Click **Next** to continue.

5.3.3.4 Capturing File Sets

Capturing file sets takes the file set definitions generated from introspection, archives these file sets into one or more zip (or other raw) files and stores the resulting files in the shared area of the catalog.

For the capture to succeed, some plug-ins have specific requirements for the reference system's running state. Table 5–2 lists the preconditions for the products supported by Oracle Virtual Assembly Builder.

Table 5–2 Capture Plug-in Requirements

Introspected Product	Running State Pre-Condition
Oracle WebLogic Server	Administration Server must be up and in the running state (not in the admin state). Managed Server(s) may be up or down.
Oracle Coherence*Web	Administration Server must be up and in the running state (not in the admin state). Managed Server(s) may be up or down.
Oracle Forms*Web	Administration Server must be up and in the running state (not in the admin state). Managed Server(s) may be up or down.
Oracle SOA for WebLogic Server	Administration Server must be up and in the running state (not in the admin state). Managed Server(s) may be up or down.
Oracle HTTP Server (OHS)	No requirement; Oracle HTTP Server may be up or down.
Oracle Single-Instance Database	In the introspection phase, the database can be up or down.

Table 5–2 (Cont.) Capture Plug-in Requirements

Introspected Product	Running State Pre-Condition
Oracle RAC Database	In the introspection phase, the database can be up or down.
Oracle Reports	Administration Server must be up and in the running state (not in the admin state). Managed Server(s) may be up or down.
Oracle Traffic Director	In the introspection phase, the Oracle Traffic Director application can be up or down.
Oracle Tuxedo	In the introspection phase, the Tuxedo application can be up or down.

5.3.3.4.1 Capturing File Sets Using Oracle Virtual Assembly Builder Studio Configure the file sets for the appliance in the *Configure New Appliance Templates* page. The file sets that appear are configured by the introspector plug-in (at this point they are read-only).

The introspector plug-in defines whether a file set is shared or local. It also defines whether or not:

- a file set can be edited (whether the user can change the properties of the file set)
- a shared file set can be switched to a local file set and vice-versa

These controls are independent of one another. The File Set dialog ensures the definitions of the introspector plugin are followed. For example, if a file set is not editable, but the shared state is editable (a shared File Set can be made local, etc.), all the fields except the shared/local radio buttons are read-only. However, the flags (*Editable* and *Shared Editable* respectively) are shown in the Property Inspector.

You can add file sets. For example, in Oracle HTTP Server there is a `DocumentRoot` path, and you may want to capture this file set.

File Set Details

Configure these parameters for a file set:

- Name: Enter a name for the file set.
- Root Directory: Enter a root directory.
- OS Owner and OS Group: For each file set, a you can specify an OS owner and group. The product makes no guarantees that the owner and/or group is defined in the base image. During deployment, the file set is expanded with the owner and group specified. The owner and groups defaults to "oracle."
- Exclusions: For each file set you can specify multiple locations under the root directory that should *not* be captured. The locations are relative to the root directory. The following patterns are allowed in the exclusions:
  - a literal path, for example foo/bar
  - any * in a trailing file/directory name, for example:
    * foo/bar/*
    * foo/bar/*.log
    * foo/bar/tmp.*

    The difference between foo/bar and foo/bar/* is that foo/bar removes bar, whereas foo/bar/* removes everything under foo/bar, but not foo/bar itself.

Note: The base image must have the owners and groups defined.

Capture File Set

Creates a directory where you can capture the file set definition in this file set. The file set definition is the set of instructions used to build a file set. Both local and remote file sets can be captured if the file system type allows it. See "File System Type".

Shared File Sets

You can configure each file set as either shared or local. If shared, users may or may not decide to capture the file set. In some cases a file set may not be allowed to migrate from local to shared or vice versa. This is defined by the introspector plug-in that created the initial file set and Oracle Assembly Builder Studio follows that setting.

You can also specify mount options for a shared file set. This occurs only in the property inspector view during deployment plan editing.

File System Type

You can specify the free space size for a given file set. Each file set can have a defined free space. This value is set on a file set by file set basis. For local file sets, you can also elect to not capture the file set, and create an empty space on the VM.

Select the file system type. For local file sets, valid choices are:

- Linux
  - Can capture a file set
  - Can define free space

For Shared File Sets, valid choices are:

- NFS
  - Can capture a file set
  - Cannot define free space
- RAW
  - Cannot capture a file set
  - Can define free space
- Native
  - Can capture a file set
  - Can define free space

Define Free Space for a File Set

Define a free space for the given file set. Select Megabyte, Gigabyte, or Percent for the free space unit, and enter a value.

The free space unit can only be defined as Percent if the file set is captured. A Native file system type cannot use Percent because Native can never capture a file set.

Validate Base Image

In the *Validate Base Image* page, you validate that the base image for each appliance meets minimum requirements. The page displays the results of validation. A

successful validation allow you to continue. An unsuccessful validation specifies the missing requirements.

Summary

The *Summary* page displays a summary of the template that you will create. Click **Finish** to start the template creation.

> **Note:** Creating templates can be time and resource (disk and network I/O) intensive. Depending on how many templates are being created this could take more than an hour to complete. Verify the actions outlined, and click **Finish** to start the process.

To accurately get the progress of a particular task, use the task log. To view a task log for a particular task, select the task from the Task viewer and click the **View Task Log** toolbar button on the Tasks navigator.

5.3.3.4.2 Capturing File Sets Using abctl The `introspect*` commands in `abctl` currently capture file sets at the end of introspection by default. This can be overridden with a flag to allow separate capture of file sets via the `captureFileSets` command.

`abctl` provides both local and remote file set capture capability. For remote file set capture, the Oracle Virtual Assembly Builder host must have SSH access to the subject machine.

Here are two examples:

Example 5–7 Capture File Sets for Oracle HTTP Server Remotely

```
$ ./abctl captureFileSets -name myOHS -remoteHost myReferenceSystemHost
-remoteUser abdemo
```

Example 5–8 Capture File Sets for Oracle WebLogic Server Locally

```
$ ./abctl captureFileSets -name myWLS -force
```

For more information see Appendix A, "Command Line Reference", which contains the details of the parameters that can be passed into the command along with a sample output of the command.

The *Summary* window lists all the templates that will be created after you click **Finish**. It also shows a warning that creating templates can take some time.

5.3.3.5 Creating Templates Using abctl

Example 5–9 through Example 5–10 are `createTemplate` command examples:

Example 5–9 create Oracle VM Guest OS template for Oracle WebLogic Server

```
$ ./abctl createTemplate -name myWLS -platform OVM
```

Example 5–10 create Oracle VM Guest OS template for OHS

```
$ ./abctl createTemplate -name myOhs -platform OVM -baseImage
/private/baseImage/OVM/OEL/System.img
```

For more information see Appendix A, "Command Line Reference", which contains the details of the parameters that can be passed into the command.

5.3.4 Editing an Assembly Using Oracle Virtual Assembly Builder Studio

This section describes how to edit an assembly, using Oracle Virtual Assembly Builder Studio. After creating an assembly, you may need to edit the assembly before it can be deployed to create connections, and optionally, to make other changes.

Note that once you have created an assembly archive you cannot edit the assembly (that is, create connections, or modify properties of appliances contained in the assembly, and so on).

5.3.4.1 Connecting Appliances

Managing connection configuration includes the connecting of inputs to outputs, setting or changing the property values of inputs and outputs (such as JDBC connection strings), and creating external resource appliances for those outputs that connect to components not deployed as part of the assembly.

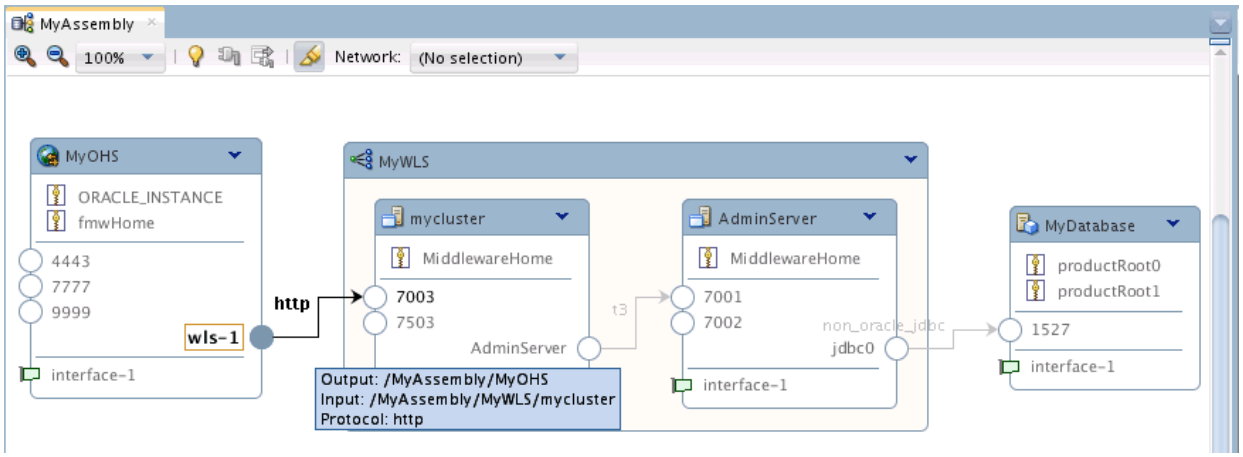
Defining a connection means generally to connect an output to an input and set/modify the properties of the input and output as necessary. The "connect output to input" step can be achieved by either connecting an output of an appliance to an input of another or connecting an output of an appliance to the input of an external resource appliance.

All input/output properties that require values must have values set and all outputs must be connected to either an input or an external resource before you can create an assembly archive. Inputs do not need to be connected. All other properties may also need to be set/modified according to the environment that will be in place at deployment time.

For example:

- **Configure Web server port forwarding**: select an output on *myohs* and connect it to a Managed Server input on the *mywls* assembly by drawing an arrow between the two.
- **Specify JDBC connect strings for each JDBC connection**: open the JDBC output of an Oracle WebLogic Server assembly by selecting it.

Figure 5–4 Connecting Appliances



This graphic shows how to connect appliances.

****

For JDBC connections, you either create external resources or introspect the database, then make the connection between the JDBC output of Oracle WebLogic Server to the external resource or the introspected database appliance.

Each of the JDBC connections has a different description. Use that description to figure out which JDBC database schema to connect each to. For each of the JDBC entries, you can look at the `original-url`, and then copy the `hostname` and `global-db-name`, into *mydb*.

For example: in `jdbc:oracle:thin:@machine999.example.com:1521/orcl`, the `hostname` is machine999.example.com, the `port` 1521, and the `global-db-name` orcl.

Also specify the `global-db-name` and `port` as properties of the external database resource input, and the host as a property of the database appliance itself.

The port is a property of the external database resource input. The hostname is the only property that belongs to the database external resource itself.

Note: Appliance-specific connection information is described in Appendix B, "Oracle Virtual Assembly Builder Introspection Plug-ins"

5.3.4.2 Auto-wiring

You can enable or disable auto-wire suggestions on the editor toolbar. If disabled, then no suggestions are shown. If enabled then suggestions are shown in two different ways.

The Suggest Wiring Toolbar Button only shows the suggested wiring when you select an output. It shows all the valid connections to different valid inputs. A valid input is the one which follows the same protocol as the output. Oracle Virtual Assembly Builder provides only one suggestion for a given output. The suggestions appear as soon as the output and input both exist on the canvas, after a drag and drop operation for example.

To accept a suggestion, you can right-click on the output, input, or suggestion and select **Accept Suggestion**. Suggestions only appear for unbound outputs.

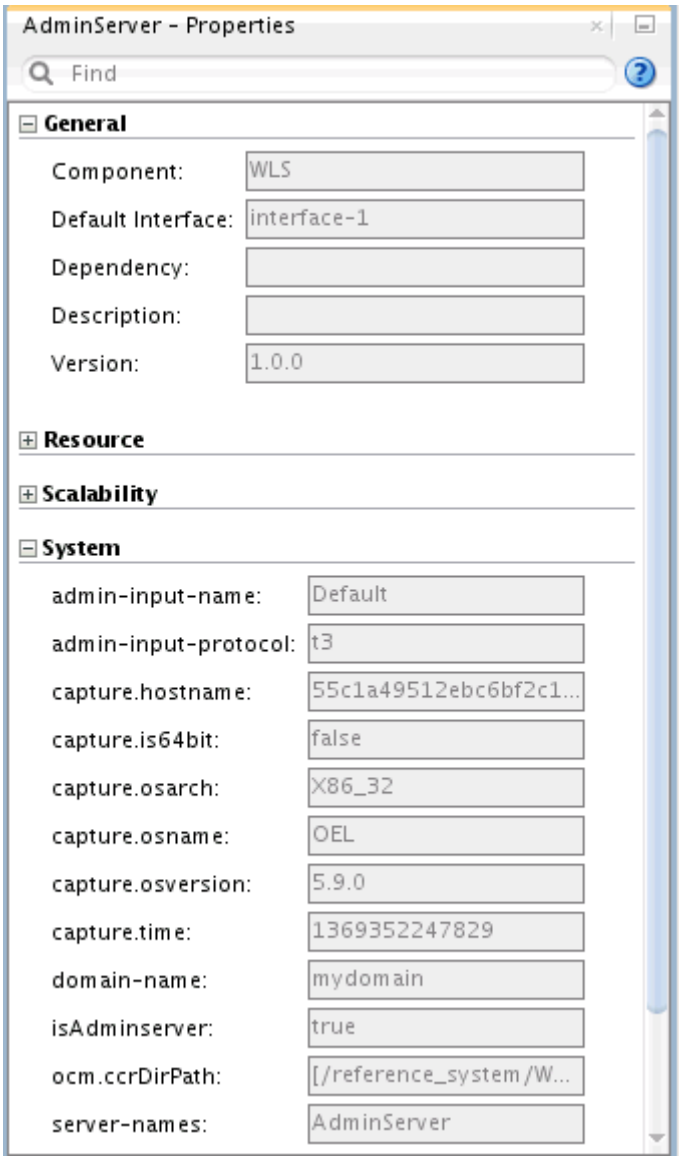
If you select an unbound output, then one dashed wire is drawn for each potential legal connection. Each potential legal connection is colored gray but if a suggestion exists, it will still be in the list as black. When you select a suggested wiring and right click on it, the **Accept** option is presented. If you click on the **Accept** button, a new connection is created (bound).

5.3.4.3 Property Inspector

You may not need to make changes to properties if the values from the reference system are appropriate. If required, make changes using the property inspector.

The property inspector (Figure 5–5) displays the property values. Properties are grouped in the property inspector during assembly editing and deployment plan editing. Set the properties as required.

Figure 5–5 Property Inspector



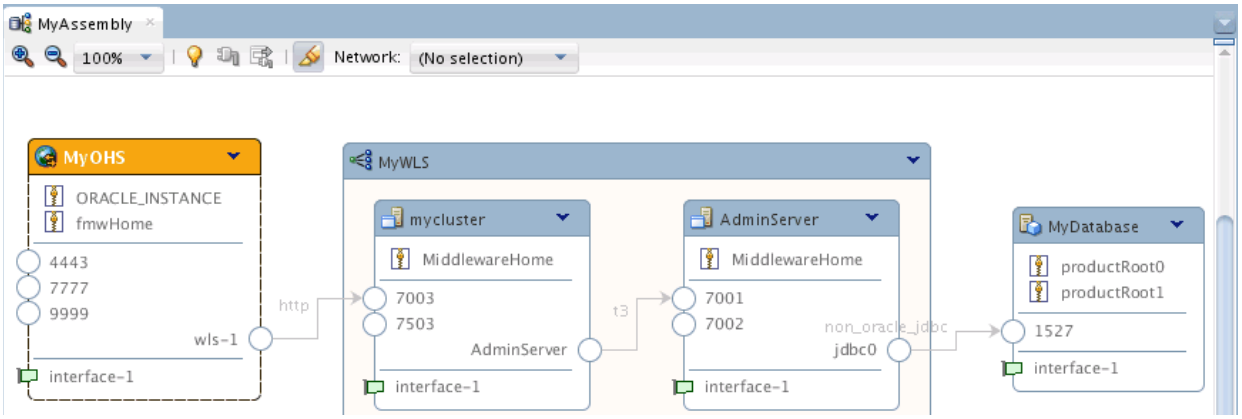
This graphics displays the Property Inspector, which is described in the surrounding text.

****

5.3.4.4 Structure Pane

The Structure Pane is populated whenever you select an element, such as an assembly, Deployer artifacts such as targets, assembly instances, and appliance instances (Figure 5–6). When you select an assembly, the Structure Pane (Figure 5–7) shows the structure of that assembly, including all appliances as well as those appliances' inputs, outputs and network interfaces. Selecting an item on the structure pane populates the property inspector with the properties scoped to that specific selection.

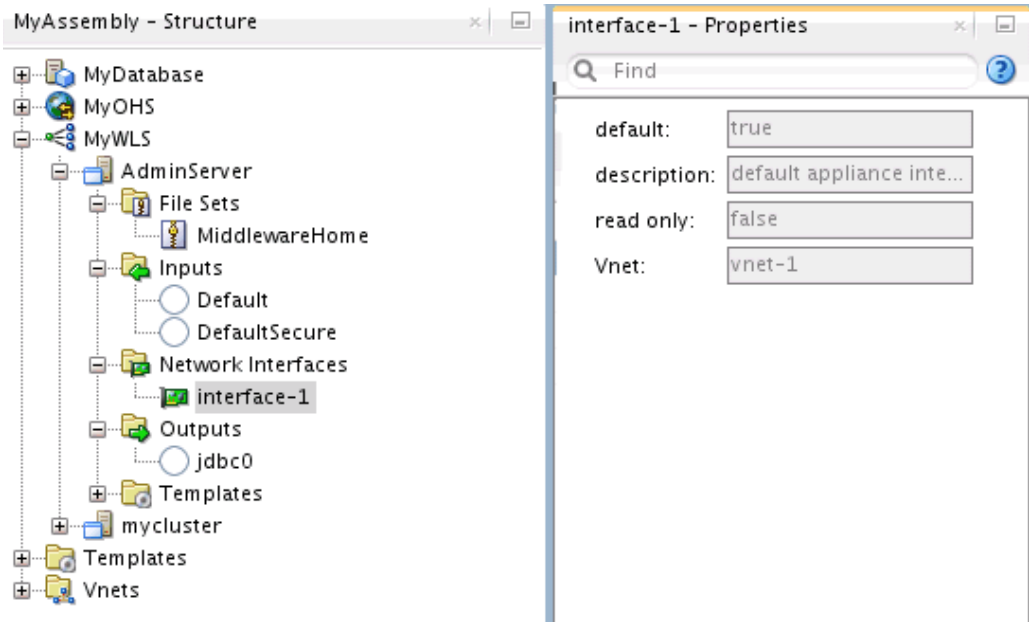
Figure 5–6 Selecting an Element



This graphic shows the result of selecting an element in the Structure Pane.

****

Figure 5–7 Structure Pane



This graphics displays the Structure Pane, which is described in the surrounding text.

****

5.3.4.5 Task Viewer

A task viewer shows the a description and status of all asychronous operations, or *tasks*, you have launched from the same Oracle Virtual Assembly Builder Studio session. Each task is displayed in the task viewer, a dockable pane positioned at the lower middle of the main frame.

During their lifetime, tasks progress through a sequence of states which you can observe in the viewer. Task states are described in Table 5–3.

Table 5–3 Task States

State	Icon or progress bar	Description
Pending	Clock icon	Initial state. Not observable for tasks that run immediately when created.
Running	Active progress bar	Entered from the PENDING state.
Cancelling	Active progress bar with the word “Cancelling…” in background	Entered from the RUNNING or PENDING state. May not be observable if task responds very rapidly to cancellation.
Cancelled	Red circle with icon	Final state. Entered from the CANCELLING state. May not be entered if task does not respond to cancellation, that is, the task may succeed or fail after an attempted cancellation.
Failed	“X” on circular red background	Final state. Entered from the RUNNING or CANCELLING state.
Succeeded	Green check mark	Final state. Entered from the RUNNING or CANCELLING state.

Table 5–4 describes the information displayed for each task.

Table 5–4 Task Viewer

State	Description
Description	A logical description of the task in progress. For example "Create template for ohs64."
Status	The current state of the task. This column displays the icon or progress bar as described in Table 5–3. If the task has a deterministic number of steps, or a percentage of progress is available, the progress bar represents the completed steps or percentage.
Started	The local time at which the task entered the RUNNING state.
Elapsed	The total time elapsed since the task entered the RUNNING state.
Completed	The local time at which the task reached a final state (SUCCEEDED, FAILED, or CANCELLED).

5.3.4.5.1 Viewing a Log of the Tasks Actions You can view a log of the task’s actions and progress by selecting the task (row) and clicking the “text” icon in the toolbar. Viewing the messages opens up a text editor in the canvas area and presents the messages associated only with that task. You can also open the task’s log by double clicking the task’s row in the table.

5.3.4.5.2 Filtering Tasks You can filter the list of displayed tasks based on their state. You can independently hide tasks in the , CANCELLED, SUCCEEDED and/or FAILED states using the latching toolbar buttons. If hidden, the task viewer does not show any tasks in that state.

You cannot filter tasks in the PENDING, RUNNING, or CANCELLING states.

5.3.4.5.3 Deleting a Task You can delete any task in a final state by selecting the task (row) and clicking the red “X” icon in the toolbar, or using the delete key.

5.3.4.5.4 Cancelling Tasks You can attempt to cancel a task by selecting the task (row). If the task supports cancellation, you can then click the "stop" (red block) icon in the toolbar. If the task does not support cancellation, the stop icon is disabled.

The type of tasks you can cancel include:

- Create assembly archive
- Capture file set (both local and remote)
- Create OEL template
- Import (both from directory and from assembly archive)

5.3.4.6 File Set Definition Editor

You can add, edit, and delete file sets from the structure pane. To perform a file set definition action, right click on the file set, or the file set folder for a given appliance in the Structure pane.

You can click on a file set to view the properties in the property inspector. You can right click a appliance and select **New > File Set** to create a file set definition. You can also right click on an individual file set to select the **Open** or **Delete** options. If the file set is editable, you will be able to modify properties in the dialog. If not, the properties will be read only.

To create, edit or delete a file set definition:

- The top-most parent assembly for the appliance must not have an assembly archive.
- If the appliance is shared, (that is, its binaries are used across multiple assemblies) and no containing top-level assembly has an assembly archive.

If the appliance already contains a template or has already had its file sets captured, but is not part of any assembly with an assembly archive, you are warned that any changes will cause that appliance's (or appliances' if shared) templates and file sets to be deleted should you accept the changes. You can also view the state of the file set (whether or not it is packaged and whether it is a system file set or a user file set).

5.3.4.7 Editing Assemblies Containing Oracle HTTP Server/Oracle Web Cache and Oracle WebLogic Server

If you have an assembly that contains Oracle HTTP Server/Oracle Web Cache and Oracle WebLogic Server with Enterprise Manager deployed, as part of deployment of Oracle HTTP Server/Oracle Web Cache "opmnctl registerInstance" is called to register that appliance with an Enterprise Manager application hosted in Oracle WebLogic Server. To enable this operation to complete successfully, you must perform the following steps while editing the assembly:

1. Define connections between Oracle HTTP Server/Oracle Web Cache's EMRegistration output and Oracle WebLogic Server.
2. Use the property inspector to set the Oracle HTTP Server/Oracle Web Cache dependency on Oracle WebLogic Server. You can do this by selecting the *Dependency* drop-down menu in the *General* section.

   Without this configuration, Enterprise Manager registration will fail because the Administration Server has not been started.
3. Verify that the WebLogic Server Administration Server has not been configured to accept only SSL connections. The "opmnctl registerInstance" does not support SSL connections to WebLogic Server.

5.3.4.8 Application Routing between Oracle HTTP Server and Oracle WebLogic Server

If the Oracle HTTP Server configuration file `mod_wl_ohs.conf` defines application routing between Oracle HTTP Server and Oracle WebLogic Server, you need to connect Oracle HTTP Server to Oracle WebLogic Server in the editor.

5.3.4.9 Creating Vnets within an Assembly

When an assembly is created, a default Vnet is created for the assembly. You can change the name and description of the default Vnet when you create the assembly. Once the assembly is created, you may also create additional Vnets other than the one created by default.

To create a new Vnet right-click the assembly and choose **New** > **Vnet**... or select the assembly and right-click the Vnets folder in the Structure Pane and choose **New Vnet**... from the context menu (Figure 5–8). Enter the name and description of the new Vnet.

Figure 5–8 Create Vnet

This graphics displays the Create Vnet dialog, which is described in the surrounding text.

****

5.3.4.10 Creating Network Interfaces within an Appliance

Each appliance must have at least one network interface and each network interface may have zero or more virtual network interfaces. When the appliance is deployed to a vServer each network interface defined on the appliance will correspond to a network interface on the vServer. To create interfaces:

- Create a physical network interface by right clicking on an appliance in the assembly editor and selecting **New Network Interface**, or by selecting the assembly in the Assembly navigator and right clicking on the appliance or the Network Interfaces folder under an appliance and selecting **New Network Interface**.
- Create virtual network interfaces by right-clicking on a physical network interface and choosing **New Virtual Interface**. You are prompted for the name and description of the new interface. Set the other properties using the Property Inspector. The result is displayed in the Structure Pane as the child of the physical network interface. Virtual network interfaces exist to model the concept of VLAN.

5.3.4.11 Binding Network Interfaces to Vnets

Assemblies and appliances have Vnets and network interfaces. Each network interface may be bound to one and only one Vnet. The deployment plan is used to associate Vnets and network interfaces with physical networks and IP addresses that exist in the deployment environment.

The binding can be changed to a different Vnet (if a different one exists) through the Property Inspector. Each logical network must be resolved at deploy time using the deployment plan (that is, each network will have properties in the deployment plan).

5.3.4.12 Binding Appliance Inputs to Network Interfaces

When configuring an appliance Input you need to specify the network interface and port on which the Input will receive (or listen for) data. You can either select a specific network interface, in which case the Input will only listen for inputs coming from that network interface or you can specify `INADDR_ANY` in which case the Input will listen for inputs coming from any network interface configured for the appliance. `INADDR_ANY` is the default.

To set the default interface on an appliance: right-click the interface in the Structure Pane and choose **Set as Default** from the context menu. The default may be either a virtual or a physical interface. Note that virtual interfaces do not appear in the Assembly Editor. They may only be selected in the Structure Pane.

5.3.4.13 Creating External Resources from an Appliance Output

Create an external resource by right-clicking on an unconnected appliance output in the assembly editor, and selecting **New External Resource**, or selecting a button on the toolbar in the canvas.

The *Create External Resources* dialog populates with all the unbound outputs, and allows you to create external resources for selected unbound outputs. You can select and choose which outputs to bind to external references. Once you click **OK** the external references appear on the canvas bound to the output. This operation is similar to the `createExternalResources` abctl command. The names for the references are auto-generated.

5.3.5 Editing an Assembly Using abctl

This section describes assembly editing operations you can perform using `abctl`.

Creating Empty Top-level Assemblies

Use the `createAssembly` command to create an empty top-level assembly. Example 5–11 shows the `createAssembly` command.

Example 5–11 createAssembly Command

```
$ ./abctl help -command createAssembly
$ ./abctl createAssembly -name myAssembly -defaultNetwork intranet
```

Adding Appliances (or WebLogic Server Assemblies) to Top-level Assemblies

Use the `addToAssembly` command to add appliances to a top-level assembly. Example 5–12 shows the `addToAssembly` command.

Example 5–12 addToAssembly Command

```
$ ./abctl help -command addToAssembly
$ ./abctl addToAssembly -name myAppliance -into myAssembly
```

Connecting Outputs to Inputs

Use the `connectEndpoints` command to creates a new connection between an output and an input. The protocols of the output and input must match, and the owners of the output and input must be part of the same assembly. Example 5–13 shows the `connectEndpoints` command.

Example 5–13 connectEndpoints Command

```
$ ./abctl help -command connectEndpoints
$ ./abctl connectEndpoints -from mySite/myOhs -fromOutput output1 -to mySite/myWls
-toInput default
```

5.3.6 Clearing Assembly Passwords

You can clear all passwords for a top-level non-atomic assembly and its child appliances and assemblies, such that all passwords must be provided within the deployment plan. This operation removes all user and template passwords (does not pertain to system property passwords since they are not reflected in the deployment plan). Optional passwords that have no value at the time the command is executed are still not required to be provided in the deployment plan. Optional passwords that have a value at the time the command is executed are set to required and their value unset.

This command does not alter any deployment plans. You must specify passwords in the deployment plan prior to deployment. If you do not provide values in the deployment plan, the deployment plan validation fails.

5.3.6.1 Clearing Assembly Passwords with Oracle Virtual Assembly Builder Studio

The Clear Assembly Passwords dialog allows you to clear the passwords from the assembly and appliance metadata. To access the Clear Assembly Passwords dialog, select **View** > **Assemblies** > **Right click an Assembly** > **Clear Assembly Passwords**. Click **OK**.

You can also perform this operation can from the Create Assembly Archive wizard.

5.3.6.2 Clearing Assembly Passwords with abctl

The `clearAssemblyPasswords` command in the abctl command-line interface allows you to clear the passwords from the assembly and appliance metadata.

Example 5–14 shows how to clear assembly passwords:

Example 5–14 Clear Assembly Passwords

```
./abctl clearAssemblyPasswords -name myWlsAssembly
```

5.3.7 Copy an Appliance or Atomic Assembly

When an appliance or atomic assembly is added to a top-level assembly, a metadata copy is created in the assembly. The new copy shares file sets and VM templates with the original. This behavior saves space and is usually desirable. In some cases, however, you may want to have a copy of an appliance or atomic assembly that does not share files and VM templates.

You can create a copy of an appliance or an atomic assembly that does not share file sets or VM templates with the original object. This operation is not allowed for

external resource appliances, appliances contained within atomic assemblies, or top-level non-atomic (user created) assemblies.

The copy is placed at the root of the catalog. Sharing of any captured file sets and/or created virtual machine templates is broken, which means new copies of those things are made as well.

5.3.7.1 Copying an Appliance or Atomic Assembly with Oracle Virtual Assembly Builder Studio

Select the appliance/atomic assembly and choose **Save As**. You can also create a copy of an appliance or assembly by exporting and then importing under a different name. The Save As dialog allows you to create a copy that does not share files sets and VM templates with the original object. In the Save As field, enter a unique name for the new appliance, and click **OK**.

5.3.7.2 Copying an Appliance or Atomic Assembly with abctl

The `copy` command in the abctl command-line interface allows you to create a copy that does not share files sets and VM templates with the original object.

Example 5–15 shows how to copy an appliance:

Example 5–15 Copy an Appliance

```
./abctl copy -name mySite/myWls -copyTo wls2
```

5.3.8 Export Operations

The following sections describes the available export scenarios:

- Export and import (of the artifacts created by the export) from the Oracle Virtual Assembly Builder Studio Catalog to another Oracle Virtual Assembly Builder Studio Catalog. See the following sections:
  - Section 5.3.8.1, "Export an Appliance or Assembly from a Catalog"
  - Section 5.3.8.2, "Import an Appliance or Assembly to a Catalog"
- Assembly Archive copy. Publishing the assembly archive outside of Oracle Virtual Assembly Builder for someone else to consume. For example, you create an assembly archive and publish it within an enterprise for others to download and deploy.
  - Section 5.3.8.3, "Exporting (Copying) an Assembly Archive"
- Importing an assembly archive from the Deployer repository into the Oracle Virtual Assembly Builder Studio catalog (import command, or Import from Studio).
  - Section 5.3.8.4, "Importing an Assembly Archive to a Catalog"

5.3.8.1 Export an Appliance or Assembly from a Catalog

This section describes how to export an appliance or assembly from a catalog, using Oracle Virtual Assembly Builder Studio, or `abctl`.

To copy an appliance or assembly from one catalog to another, you must use Oracle Virtual Assembly Builder's export and import functionality.

Note: Manual copying of disk files from one catalog to another is not supported and will not work.

5.3.8.1.1 Exporting an Appliance or Assembly from a Catalog Using Oracle Virtual Assembly Builder Studio

Access the Export dialog box (Figure 5–9) to export an appliance or assembly from a catalog by selecting **File** > **Export**, or by right-clicking the assembly in the Assemblies navigator and select **Export**.

Enter the following information:

- *Name*: this field pre-populates with the name of the appliance or assembly that you selected for export.
- *Directory*: browse to and select or enter the name of the directory of the location of the export. This directory must be empty and will be created if it does not exist.
- *Working Directory*: optionally, specify the path to a working directory.
- *Metadata Only*: check this checkbox to export only metadata (and not the associated templates or file sets).

Click **OK**.

Figure 5–9 Exporting an Appliance or Assembly from a Catalog

This graphic displays the Export window, which is described in the surrounding text.

****

5.3.8.1.2 Exporting an Appliance or Assembly from a Catalog Using abctl

Use the `export` command to export an assembly, or assembly metadata. Example 5–16 shows the `export` command for exporting metadata, and associated templates and file sets. Example 5–17 shows exporting metadata only.

Example 5–16 export Command

```
$ ./abctl help -command export
$ ./abctl export -name myOhs -toDir /tmp/myOhs.export/
(some progress messages)
Successfully exported to /tmp/myOhs.export/.
```

Example 5–17 export Command (Metadata Only)

```
$ ./abctl export -name myOhs -to /tmp/myOhs.export/ -metadataOnly
(some progress messages)
Successfully exported to /tmp/myOhs.export/.
```

For more information see Appendix A, "Command Line Reference", which contains the details of the parameters that can be passed into the command along with a sample output of the command.

5.3.8.2 Import an Appliance or Assembly to a Catalog

This section describes how to import an appliance, assembly, or assembly archive using Oracle Virtual Assembly Builder Studio, or `abctl`.

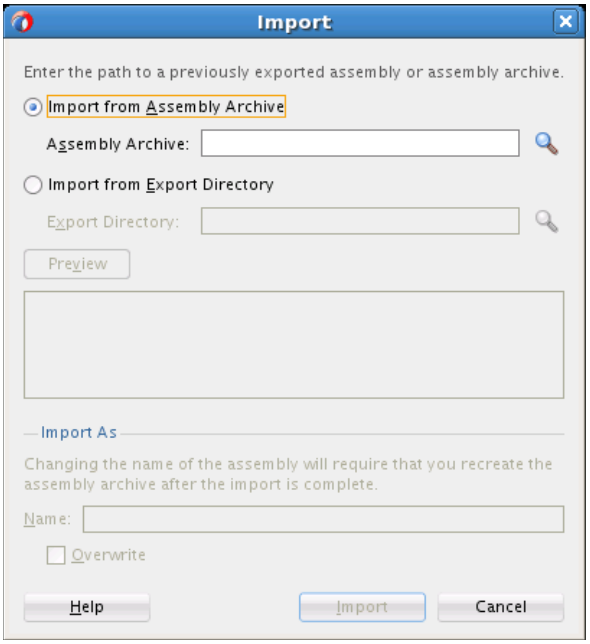
To copy an appliance or assembly from one catalog to another, you must use Oracle Virtual Assembly Builder's export and import functionality.

5.3.8.2.1 Importing Using Oracle Virtual Assembly Builder Studio Access the Import dialog box (Figure 5–10) to import an appliance or assembly to a Catalog by selecting **File** > **Import**. Enter the following information:

- *Path*: browse to and select or enter the name of the directory of the assembly or appliance which was exported. The path and associated assembly appear in the window.
- *Overwrite*: check this checkbox to specify that any existing metadata and associated file sets and templates are overridden. This is to correct a case of name collision. Overriding an existing appliance can only be done if the existing appliance can be removed.

Click **OK**.

Figure 5–10 Importing an Assembly Archive



This graphic displays the Import window, which is described in the surrounding text.

****

5.3.8.2.2 Importing an Appliance or Assembly Using abctl Use the `import` command to import (into the target catalog) the content of one or more files containing a sparse copy of exported metadata and associated file set and templates.

A new entry is created in the target catalog. If there is a name collision (for example, the import command attempts to create 'mySite', and the catalog already has 'mySite'), the operation will fail.

Example 5–18 shows the `import` command from a directory where you previously ran the export command:

Example 5–18 import Command from a Directory

```
$ ./abctl help -command import
$ ./abctl import -from /tmp/myOhs.export/
Successfully imported myOhs to /example/ab_home/catalog.
```

Example 5–19 shows the `import` command from an assembly archive:

Example 5–19 import Command from an Assembly Archive

```
$ ./abctl help -command import
$ ./abctl import -from /tmp/myOhs.ova
Successfully imported myOhs to /example/ab_home/catalog.
```

For more information see Appendix A, "Command Line Reference", which contains the details of the parameters that can be passed into the command.

5.3.8.3 Exporting (Copying) an Assembly Archive

Assembly archives in the Oracle Virtual Assembly Builder Studio catalog are stored in the `$AB_INSTANCE/archives` directory. You can copy or use FTP to transfer the assembly archives from this directory without the need for any export utility. Typically you will copy the assembly archives to the *Deployments* navigator managed by Deployer.

5.3.8.4 Importing an Assembly Archive to a Catalog

Importing an assembly archive allows you to import an assembly archive template file from a disk location into the local catalog to allow editing of the assembly appliances. Importing creates a metadata structure, with file set and template artifacts that are identical to the original catalog.

To import an assembly archive use the `downloadAssemblyArchive` command in `abctl`:

Example 5–20 downloadAssemblyArchive Command

```
$ ./abctl help -command downloadAssemblyArchive
$ ./abctl downloadAssemblyArchive -name MyAssembly -version 1 -fileName
RenamedAssembly.ova
```

For more information see Appendix A, "Command Line Reference", which contains the details of the parameters that can be passed into the command.

5.3.9 Rename an External Resource

When you create an external resource, the name of the resource is selected automatically. This name selection is performed in the infrastructure and supports

reuse in that functionally identical existing external resources are reused where possible.

You can the rename any external resource. The operation is not supported for other types of catalog objects. Only external resources can be renamed.

5.3.9.1 Renaming an External Resource with Oracle Virtual Assembly Builder Studio

The Rename dialog allows you to rename an external resource. To access the dialog, select **File** > **Rename** from the main menu, or select Rename from the context menu of the external resources.

In the Rename field, enter a new name for the external resource, and click **OK**.

5.3.9.2 Renaming an External Resource with abctl

The `renameExternalResource` command in the abctl command-line interface allows you to renames a specified external resource.

Example 5–21 shows how to rename an external resource:

Example 5–21 Rename an External Resource

```
 ./abctl abctl renameExternalResource -assembly mySite -currentName jdbc0 -newName
myDatabase
```

5.3.10 Creating an Assembly Archive

You can create an assembly archive for a given assembly. An assembly archive contains metadata about the assembly and assembly templates that are used to instantiate an instance of the assembly in a virtualized environment.

When the assembly archive is successfully created, the assembly is locked. At this point you cannot make changes to the assembly without explicitly deleting the assembly archive.

5.3.10.1 Creating Assembly Archives Using Oracle Virtual Assembly Builder Studio

To create an assembly archive for an assembly:

1. Select **View** > **Assemblies** to view the available assemblies.
2. Right click an assembly and select **New** > **OVM Assembly Archive** to create an assembly archive for the assembly.
3. In the *Welcome* page, click **Next**.
4. In the *Create Assembly Archive* page, do the following:
    1. Select the Compress Assembly Archive checkbox if you want to create and compress an assembly archive.
    2. Select the Sign Assembly Archive checkbox to sign the archive and provide the signing information.
    3. Click **Next**.
5. In the *Clear Assembly Passwords* page, select whether to clear all the passwords from the assembly and appliance metadata. Clearing passwords does not impact any existing appliance templates.

WARNING: Once passwords have been cleared they cannot be recovered.

6. In the *Recreate Existing Appliance Templates* page, select those existing appliance templates that need to be recreated by checking **Recreate**. You cannot recreate a template if the template is already included in a different assembly archive.
7. Perform the required configuration on the remaining pages (*Configure Base Image Options*, *Configure New Appliance Templates*, *Validate Base Images*, and *Summary*) as described in Section 5.3.3, "Creating Templates for an Appliance or an Assembly".

5.3.10.2 Creating Assembly Archives Using abctl

You can use `abctl` to create an assembly archive for the named top-level assembly. This command can only be invoked on a top-level assembly. Additionally, you must have previously templated all the subappliances within the assembly using the `createTemplate` command.

Use the `-platform` option to specify the platform as Oracle VM.

Example 5–22 is a `createAssemblyArchive` command example:

Example 5–22 Create Assembly Archive

```
$ ./abctl createAssemblyArchive -name myWlsAssembly -platform OVM -nocompress
```

For more information see Appendix A, "Command Line Reference", which contains the details of the parameters that can be passed into the command.

5.4 Operations Related to Deployment

This section details how you will use Oracle Virtual Assembly Builder Studio or the `abctl` command line utility to perform operations related to deployment.

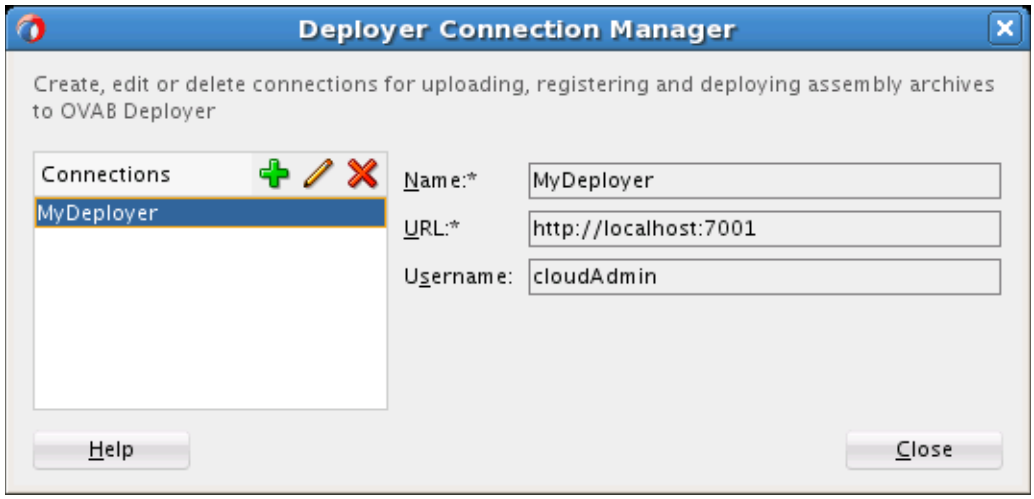
- Section 5.4.1, "Managing Deployer Connections"
- Section 5.4.2, "Configuring Targets"
- Section 5.4.3, "Creating a Deployment Plan"
- Section 5.4.5, "Checking Target Resources"
- Section 5.4.6, "Uploading an Assembly Archive to the Deployer Repository"
- Section 5.4.7, "Uploading Assembly Resources"
- Section 5.4.8, "Downloading an Assembly Archive from the Deployer Repository"
- Section 5.4.9, "Registering an Assembly Archive to a Target"
- Section 5.4.10, "Deploying an Assembly Instance"
- Section 5.4.11, "Performing One-Step Deployment"
- Section 5.4.12, "Stopping an Assembly Instance"
- Section 5.4.13, "Starting or Restarting an Assembly Instance"
- Section 5.4.14, "Scale Appliance"
- Section 5.4.15, "Undeploying an Assembly Instance"
- Section 5.4.16, "Unregistering an Assembly Archive from a Target"

- Section 5.4.17, "Deleting an Assembly Archive from the Deployer Repository"
- Section 5.4.18, "Interacting with EM Software Library"

5.4.1 Managing Deployer Connections

You can manage connections to the Deployer by selecting **Tools** > **Manage Deployer Connections**. The Deployer Connection Manager dialog appears (Figure 5–11), which allows you to create, edit, or delete connections for uploading, registering and deploying assembly archives to the Deployer.

Figure 5–11 Deployer Connection Manager



This graphic displays the Deployer Connection Manager, which is described in the surrounding text.

****

To create a connection to the Deployer:

1. Select the create icon.
2. Configure the following connection properties:
   - Name: enter a name for the Deployer connection.
   - URL: enter a URL for the Deployer connection.
   - Username: enter a username for accessing the Deployer.
   - Password: enter the password for accessing the Deployer.
3. Click **Test Connection** to verify that you can successfully connect.

   If you test a Deployer connection that uses HTTPS (TLS/SSL), additional steps may be required. If the Deployer trust store determines the connection to be trusted, then nothing is required. If the Deployer trust store does not trust the connection, you are prompted with the chain of certificates returned by the Deployer and asked if you want to add them to the Deployer trust store. To do so, you must provide the Deployer trust store password that you specified during installation.
4. Click **OK**.

To edit a connection to the Deployer:

1. Select the connection in the Connections pane.
2. Select the edit icon.
3. Updates the connection properties as required.
4. Click **OK**.

To delete a connection to the Deployer:

1. Select the connection in the Connections pane.
2. Select the delete icon.
3. Confirm the deletion.

5.4.2 Configuring Targets

The following configuration tasks are required for Oracle VM Manager and Oracle Virtual Assembly Builder Deployer to configure deployment targets:

- configure Oracle VM Manager connection parameters
- configure Oracle Virtual Assembly Builder Deployer properties
- configure the connection from the Deployer to Oracle VM Manager
- define the connection to the Oracle VM backend endpoints
- add deployment targets in the backend

5.4.2.1 Configure Oracle VM Manager

By default, when Oracle VM Manager is installed it is setup with a secure connection (that is, HTTPS). SSL communication to Oracle VM Manager needs to be configured to use Java Secure Socket Extension (JSSE) for Oracle Virtual Assembly Builder to be able to connect to it.

To configure JSSE for Oracle VM Manager:

1. Log into the Oracle WLS administration console of Oracle VM Manager.
2. Go to **Servers** > **AdminServer** > **SSL** > **Advanced Properties**.
3. Make sure **Use JSSE SSL** is checked.

> **Note:** If a change was made for JSSE, you must restart Oracle VM Manager for it to take effect.
>
> You can use a non-SSL connection (HTTP) to Oracle VM Manager if it is enabled.

5.4.2.2 Identify Oracle VM Manager Connection Parameters

Identify the following connection properties for Oracle VM Manager:

- URL
- Username/Password
- Pool

5.4.2.3 Stop Oracle Virtual Assembly Builder Deployer

Before proceeding to the following steps ensure that the Oracle Virtual Assembly Builder Deployer is not running. Use the `ps -ef | grep` command for Java

processes if needed. If you just completed creating the Deployer domain the Deployer will not be running. The server needs to be explicitly started.

5.4.2.4 Configure Deployer Properties

To configure properties:

1. Ensure the Oracle Virtual Assembly Builder Deployer is not running.
2. Edit the `<deployer domain root>/config/fmwconfig/servers/AdminServer/mbeans/ovabconfig.properties` file and set the `ovab.webserver.url` property

   appropriately. By default it is set to "http\:localhost\:7001". The ":" is escaped by a "\". Determine the IP address of the host machine and edit the `ovab.webserver.url` appropriately. For example:

   ```
   ovab.webserver.url=http\://192.168.1.1\:7001
   ```

Note: This IP address must be reachable. If the IP is not reachable the Deployer operations associated with deployments will fail. This is the callback URL that the Oracle VM Manager uses to communicate with the Deployer.

5.4.2.5 Configure Deployer for SSL connection to Oracle VM Manager

To create an SSL connection to Oracle VM Manager, if required:

1. Install the Oracle VM Manager certificate in the Deployer:
   1. Issue the following command:

      ```
      /usr/bin/openssl s_client -connect <ovmm_host>:<port>
      ```

      For example:

      ```
      [ovab@localhost ~]$ /usr/bin/openssl s_client -connect
      ovmmhost.example.com:7002
      CONNECTED(00000003)
      depth=0 /C=US/O=Oracle/OU=OVMM/CN=weblogic
      verify error:num=18:self signed certificate
      verify return:1
      depth=0 /C=US/O=Oracle/OU=OVMM/CN=weblogic
      verify return:1
      ---
      Certificate chain
        0 s:/C=US/O=Oracle/OU=OVMM/CN=weblogic
          i:/C=US/O=Oracle/OU=OVMM/CN=weblogic
      ---
      Server certificate
      -----BEGIN CERTIFICATE-----
      tZh6pp/foLKF6tczer9CtH58gXdRKIjRsN3kYh1NUhYwdMQhwrJeWgvDzs/AFpMy
      UzEPMA0GA1UEChMGT3JhY2xlMQ0wCwYDVQQLEwRPVk1NMREwDwYDVQQDEwh3ZWJs
      tZh6pp/foLKF6tczer9CtH58gXdRKIjRsN3kYh1NUhYwdMQhwrJeWgvDzs/AFpMy
      AlVTMQ8wDQYDVQQKEwZPcmFjbGUxDTALBgNVBAsTBE9WTU0xETAPBgNVBAMTCHdl
      YmxvZ2ljMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDVoMviwZXEyGTA0/gz
      tZh6pp/foLKF6tczer9CtH58gXdRKIjRsN3kYh1NUhYwdMQhwrJeWgvDzs/AFpMy
      pawpiL2jG/FoVA4qZDgxmFSfq70Qo0x8Csm9ZRJObb1229UDlzz9i4/wOZywLT65
      tZh6pp/foLKF6tczer9CtH58gXdRKIjRsN3kYh1NUhYwdMQhwrJeWgvDzs/AFpMy
      5tZh6pp/foLKF6tczer9CtH58gXdRKIjRsN3kYh1NUhYwdMQhwrJeWgvDzs/AFpMy
      bAUGIELgtYnR/iG1RVEp4yQxSYqS9qdwoPwW/yhXspft9UwH0vNWUedzriRe5pA9
      ```

```
216UH/+/RYtZ9wdFDDdOebT1u2bHWJExsVi8
-----END CERTIFICATE-----
subject=/C=US/O=Oracle/OU=OVMM/CN=weblogic
issuer=/C=US/O=Oracle/OU=OVMM/CN=weblogic
---
No client certificate CA names sent
---
SSL handshake has read 1086 bytes and written 279 bytes
---
New, TLSv1/SSLv3, Cipher is EDH-RSA-DES-CBC3-SHA
Server public key is 1024 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
SSL-Session:
  Protocol : TLSv1
  Cipher : EDH-RSA-DES-CBC3-SHA
  Session-ID:
517AF76A8665187015EC313D4FA2BC72B06B58A11268F3D4443F86517AF76A8
  Session-ID-ctx:
  Master-Key:
517AF76A8BAC77A5C9571DA050343239C821CBCC3273CE781C64B8EF5E0137FBA93C67253D
0D0A0E915CF517AF76A8
  Key-Arg : None
  Krb5 Principal: None
  Start Time: 1367013332
  Timeout : 300 (sec)
  Verify return code: 18 (self signed certificate)
---
read:errno=0
[ovab@localhost ~]$
```

2. Copy the cert portion of the above output (from "BEGIN" to "END", inclusively) to `/tmp/ovmm.cert`:

```
-----BEGIN CERTIFICATE-----
tZh6pp/foLKF6tczer9CtH58gXdRKIjRsN3kYh1NUhYwdMQhwrJeWgvDzs/AFpMy
UzEPMA0GA1UEChMGT3JhY2xlMQ0wCwYDVQQLEwRPVk1NMREwDwYDVQQDEwh3ZWJs
tZh6pp/foLKF6tczer9CtH58gXdRKIjRsN3kYh1NUhYwdMQhwrJeWgvDzs/AFpMy
AlVTMQ8wDQYDVQQKEwZPcmFjbGUxDTALBgNVBAsTBE9WTU0xETAPBgNVBAMTCHdl
YmxvZ2ljMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDVoMviwZXEyGTA0/gz
tZh6pp/foLKF6tczer9CtH58gXdRKIjRsN3kYh1NUhYwdMQhwrJeWgvDzs/AFpMy
pawpiL2jG/FoVA4qZDgxmFSfq70Qo0x8Csm9ZRJObb1229UDlzz9i4/wOZywLT65
tZh6pp/foLKF6tczer9CtH58gXdRKIjRsN3kYh1NUhYwdMQhwrJeWgvDzs/AFpMy
5tZh6pp/foLKF6tczer9CtH58gXdRKIjRsN3kYh1NUhYwdMQhwrJeWgvDzs/AFpMy
bAUGIELgtYnR/iG1RVEp4yQxSYqS9qdwoPwW/yhXspft9UwH0vNWUedzriRe5pA9
216UH/+/RYtZ9wdFDDdOebT1u2bHWJExsVi8
-----END CERTIFICATE-----
```

3. Issue the following command:

Note: This example shows the steps for when the demo truststore is in use by the Deployer. If using a custom truststore, substitute the appropriate truststore path and pass phrase.

```
keytool -import -v -trustcacerts -alias ovmm_cert -file /tmp/ovmm.cert
-keystore <deployer's weblogic home>/server/lib/DemoTrust.jks
```

Note: You are prompted for a password. It is: DemoTrustKeyStorePassPhrase.

You are prompted to trust the certificate. Enter yes.

Here is a sample execution:

```
[ovab@localhost ~]$ keytool -import -v -trustcacerts -alias ovmm_cert -
file /tmp/ovmm.cert -keystore
/home/ovab/Oracle/Middleware/Oracle_Home/wlserver/server/lib/DemoTrust.jks
Enter keystore password:
Owner: CN=weblogic, OU=OVMM, O=Oracle, C=US
Issuer: CN=weblogic, OU=OVMM, O=Oracle, C=US
Serial number: 50866375
Valid from: Tue Oct 23 05:29:25 EDT 2012 until: Fri Oct 21 05:29:25 EDT
2022
Certificate fingerprints:
  MD5: BE:75:9D:CF:5F:21:6E:40:6E:3D:1E:A0:37:BE:22:C2
  SHA1: 3C:69:CA:BA:58:DF:4D:AB:CF:2F:53:84:D9:B8:9D:40:CA:A3:F8:B5
  SHA256:
EA:0A:49:A0:C3:15:31:C5:5D:4A:3F:2D:AD:3A:18:C4:CC:55:C2:11:67:93:A3:C2:F5
:68:1F:BD:2F:21:63:BC
  Signature algorithm name: SHA1withRSA
  Version: 3
Trust this certificate? [no]: yes
Certificate was added to keystore
[Storing
/home/ovab/Oracle/Middleware/Oracle_Home/wlserver/server/lib/DemoTrust.jks
]
[ovab@localhost ~]$
```

5.4.2.6 Start Oracle Virtual Assembly Builder Deployer

Start the Oracle WebLogic Server hosting the Deployer.

Note: Wait for the server to reach the "RUNNING" mode.

5.4.2.7 Connection URL

Oracle requires that you configure your target connections for Oracle VM 3.2.1 with HTTP/HTTPS protocol.

To configure with HTTP, specify a URL of the form "http://their-ovm-host:7001".

5.4.2.8 Connection Credentials

To configure connection to Oracle VM Manager:

1. The HTTP/HTTPS port is used to connect to Oracle VM Manager. If the Oracle VM Manager protocol HTTPS is used (which will be the usual, since OVMM disables HTTP by default), you must import the Oracle VM Manager certificate into the Deployer's trust store prior to creating a target.

   See Section 5.4.2.5, "Configure Deployer for SSL connection to Oracle VM Manager" for information on importing the certificate into the trust store.

2. Create the target using the `createTarget` command in abctl.

This operation, which can only be performed by the CloudAdmin, defines the connection information, and, depending on the backend type, user credentials for the backend. Specify `ovm.vmmversion`=3.2 and Oracle VM credentials here.

5.4.2.9 Examples

Example 5–23 shows how to create a target for Oracle VM:

Example 5–23 Create Target for Oracle VM

```
./abctl createTarget -name slcTarget_http -type ovm -properties ovm.poolName=ab_
ovm_30_stand_alone_pool ovm.vmOperationTimeout=3600 ovm.vmmversion=3.2
ovm.user=admin ovm.pwd ovm.url=https://example.com:7002 -connectionName
localDeployer
```

Example 5–24 shows how to add users to a target for Oracle VM:

Example 5–24 Add Users to a Target for Oracle VM

```
./abctl addTargetUser -user Username -target Targetname
```

5.4.3 Creating a Deployment Plan

A deployment plan allows you to override property values defined in the metadata of an assembly so you can customize each deployment of the assembly. The plan is applied when the assembly is deployed. Only top-level assemblies can have deployment plans.

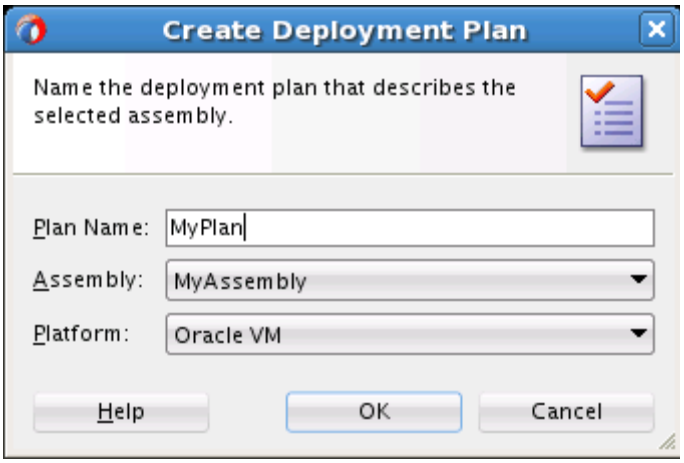
5.4.3.1 Create a Deployment Plan Using Oracle Virtual Assembly Builder Studio

Create a deployment plan:

1. In the Deployment Plans navigator, right-click an assembly and select **New Deployment Plan**.

   The *Create Deployment Plan* wizard appears (Figure 5–12).

Figure 5–12 Create Deployment Plan



This graphic displays the Create Deployment Plan window, which is described in the surrounding text.

****

2. Enter the name for the deployment plan.
3. Select the associated assembly from the Assembly drop-down menu.
4. Click **OK**. The Deployment Plan editor opens.

5.4.4 Editing a Deployment Plan

The Deployment Plan Editor displays a read-only view of the assembly. This view is useful as an overview and for selecting items so that their property values may be overridden.

5.4.4.1 Required Views

Deployment plan editing makes use of both the Structure Pane and the Property Inspector. You should ensure both views are visible.

1. Open the Structure Pane by selecting **View** > **Structure**.
2. Open the Property Inspector by selecting **View** > **Property Inspector**.

5.4.4.2 Edit a New Deployment Plan

When you create a new deployment plan, the Deployment Plan editor opens automatically.

5.4.4.3 Edit an Existing Deployment Plan

Edit an existing deployment plan:

1. In the Deployment Plans navigator, right-click the plan and select **Open**, or double-click the plan icon in the Deployment Plans navigator.

   You can also edit a deployment plan by selecting the deployment plan in the Deployment Plan navigator and selecting the appliances/network interfaces in the structure pane. The properties of any element selected in the structure pane are reflected in the property inspector.

5.4.4.4 Selecting items in the Deployment Plan Editor

To override the properties of an appliance, input, output or other item, you must first select the item.

To select an item in the Deployment Plan editor, click on it.

5.4.4.5 Selecting items in the Structure Pane

When populated, the Structure Pane shows additional details of the plan.

Populate the Structure Pane:

1. In the Deployment Plans navigator, select the plan.

   The Structure Pane populates with the assembly structure. To select an item in the populated Structure Pane, click on it.

   Selecting any item in the Deployment Plan editor also populates the Structure Pane.

5.4.4.6 Overriding Property Values

The properties of the selected item are displayed in the Property Inspector.

To override a value:

1. In the property inspector, type a new value in the field.

   A blue bullet is displayed next to overridden values.

5.4.4.7 Remove an Override Value

To remove an override:

1. In the Property Inspector, click the downward-pointing arrow (chevron) to the right of the property value.
2. From the pop-up menu, select **Reset to**...

   The value shown in the **Reset to**... menu item is always the original value specified in the assembly metadata. The override is removed.

5.4.4.8 Ability to Set Appliance 'target' Count to Zero (Zero-Count Appliances)

During deployment plan editing, you can edit the target scale to zero in the deployment plan to initially deploy an assembly with zero appliance instances.

In subsequent scaling operations you can add appliance instances to those appliances that are part of the assembly configuration but were initially "deployed" with a zero instance count.

5.4.4.9 Synthetic Properties

Synthetic properties are properties needed to successfully deploy an assembly, that Oracle Virtual Assembly Builder adds to appliances, networks and network interfaces. These properties include, scaling, network, resource requirements and several more.

You may override the scalability properties of an appliance. Doing so may change the number of potential appliance instances that can exist when the assembly is deployed. Each of these potential appliance instances may require its own network settings including hostname, IP address, MAC address, and netmask properties. The Property Inspector automatically displays the appropriate number of instance-specific network properties. When the scalability values are changed, the Property Inspector automatically adjusts the number of network properties to match the number of appliances instances that may potentially exist.

5.4.4.9.1 Appliance Properties To display the network properties of an appliance:

1. In the Structure pane or Deployment Plan editor, click to select the appliance.
2. In the Property Inspector, click to open the Network category.

The *default-gateway*, *dns-domains*, and *dns-servers* property values are required unless all network interfaces of the appliance are configured to use DHCP. These values are shared by all instances of the appliance.

The *hostname.0* property is required. If the appliance is scalable, the Property Inspector will display an appropriate number of properties (*hostname.0*, *hostname.1*, etc.) All values are required.

5.4.4.9.2 Network Interface Properties To display the Deployment Plan properties of a network interface, click on the network interface in the Deployment Plan editor or expand the Network Interfaces folder in the Structure pane and click the network interface within the folder.

Setting the *usedhcp* property of the Interface to **true** asserts that these network properties will be automatically configured in the deployment target environment. When *usedhcp* is **true**, the Property Inspector displays previously-established values of

the other network interface properties, but does not permit the values of these properties to be modified.

If *usedhcp* is false, the *ip_address.0*, *mac_address.0*, and *netmask* properties are required. If the appliance is scalable, the Property Inspector will display an appropriate number of address properties (*ip_address.0*, *ip_address.1*, ..., *mac_address.0*, *mac_address.1*, ...) and all values are required.

5.4.4.9.3 Vnet Properties Vnets are not displayed in the Deployment Plan editor so cannot be selected there. To display the Deployment Plan properties of a Vnet, expand the Vnets folder in the Structure pane and select the Vnet.

The *is_private* property specifies that the network should be automatically configured as a high-performance network for internal use within the deployed Assembly. This value must be **false** if the deployment target environment does not provide the necessary platform support for high-performance private networks. Set this value to **false** when deploying directly to Oracle Virtual Machine version 3.0.

The *network_name* property specifies the name of the network defined in the deployment target environment. This property is required.

Some deployment target environments may not require that network names be unique. When network names are not unique, the optional *network_id* property can be used to uniquely specify the network defined in the deployment target environment. The value of this property is ignored when the *is_private* property is true.

5.4.4.10 Validating a Deployment Plan

To validate the deployment plan:

1. In the Deployment Plans navigator, right-click the plan and select **Validate**.

   The validation results are displayed in a dialog box.

5.4.4.11 Saving a Deployment Plan

When the deployment plan has been modified, its name is displayed in italic font in both the Deployment Plan navigator and the tab of the Deployment Plan editor.

To save the deployment plan:

1. Select **File** > **Save**.

If the File > Save menu item is not enabled, the deployment plan may not be selected. In the Deployment Plan navigator, select the plan, then select **File** > **Save**.

You can create, edit and save multiple deployment plans for a single assembly. At deployment time, you select one plan for the deployment. In this way, a single assembly may serve as the basis of multiple deployments, each with their own specific network and other property settings.

5.4.5 Checking Target Resources

You can check whether there are enough resources on the target network to deploy your assembly with a particular deployment plan. You must have already created a deployment plan.

There are four places in Oracle Virtual Assembly Builder Studio where you can validate that there are enough resources for the assembly and deployment plan combination:

- Local assembly: you can right click an assembly with an assembly archive and select **Check Available Resources**. Once selected, the user is presented a modified version of the deployment dialog. Users will need to select a target and deployment plan as well as resolve any vnet bindings before clicking on the "check resources" button. A success dialog will be shown if sufficient resources exist. Otherwise a failed dialog will be shown specifying which resources are insufficient for a deployment.
- Uploaded assembly: you can right click on an uploaded assembly archive version and select **Check Available Resources**. All other behavior is identical to the local assembly.
- Registered assembly: you can select a specific registered instance of an assembly archive and select **Check Available Resources**. You do not need to select a target.
- Deployment plan: you can select a specific deployment plan of an assembly and select **Check Available Resources**. You do not need to select a deployment plan, but must select a target.

In the Check Target Resources dialog, configure the following fields:

- Target: select the deployment target.
- Deployment Plan: select the deployment plan for the deployment.
- Vnet Mappings: map Vnets from your assembly to target networks. Any change to mappings will be saved in the selected deployment plan prior to deployment. Select the target network to map to the Vnet. The current mappings are designated by an asterisk.

5.4.6 Uploading an Assembly Archive to the Deployer Repository

You can upload an assembly to the Deployer repository using Oracle Virtual Assembly Builder Studio or `abctl`.

5.4.6.1 Uploading an Assembly using Oracle Virtual Assembly Builder Studio

To upload the assembly archive, right click on the assembly and selecting **Upload Assembly Archive** > **Deployer**.

5.4.6.2 Uploading an Assembly using abctl

To upload an assembly archive, use the `uploadAssemblyArchive` command.

Example 5–25 uploadAssemblyArchive Command

```
$ ./abctl help -command uploadAssemblyArchive
$ ./abctl uploadAssemblyArchive -fileName Path -name String [-description String]
-connectionName String
```

For more information see Appendix A, "Command Line Reference", which contains the details of the parameters that can be passed into the command.

5.4.7 Uploading Assembly Resources

Assembly resources are a collection of files used during registration and/or deployment of an assembly. You upload assembly resources in a zip format file and store them alongside a specific assembly archive version. Generally, the initial zip should contain all necessary resources, however, there is an *append* option to the upload operation that allows adding more resources. The *delete* operation removes all

resources for an assembly version by default. The *file* option allows for deleting a subset. The two options mentioned – *append* and *file* – allow for arbitrary updates.

5.4.7.1 Volume Management

The assembly resources file contain disk images which are used to create a file system template on a zfs storage device at assembly registration time and these are cloned into filesystems that are mounted on the VM at deployment time.

The resource zip contains a `disks` folder that has a set of disk images. These images are referenced by the OVF metadata for the assembly. The following is an example of the OVF Metadata. The first entry (`volume|MyVolume|content-filename`) is a reference to a disk image in the assembly resource zip and is not configurable. The other properties are details on the volume that the Deployer uses for setup and to communicate details to the vServer at deploy time.

```
<ovf:ProductSection ovf:class="com.oracle.ovab.volumes">
  <ovf:Property
    ovf:key="volume|MyVolume|content-filename"
    ovf:type="string" ovf:userConfigurable="false"
    ovf:value="myIsoFileName"/>
  <ovf:Property
    ovf:key="volume|MyVolume|mount-options"
    ovf:type="string"
    ovf:userConfigurable="true" ovf:value=""/>
  <ovf:Property
    ovf:key="volume|MyVolume|mount-target"
    ovf:type="string"
    ovf:userConfigurable="true" ovf:value=""/>
  <ovf:Property
    ovf:key="volume|MyVolume|nfs-type"
    ovf:type="string"
    ovf:userConfigurable="true" ovf:value="nfs"/>
  <ovf:Property
    ovf:key="volume|MyVolume|quota-size"
    ovf:type="sint32"
    ovf:userConfigurable="true" ovf:value="10"/>
  <ovf:Property
    ovf:key="volume|MyVolume|quota-unit"
    ovf:type="string"
    ovf:userConfigurable="true" ovf:value="GB"/>
...
```

Based on these definitions, the Deployer uses the disk images from the assembly resource to create a filesystem in a ZFS storage device at assembly registration time. Later, at deployment time, this filesystem is cloned for specific use by individual vServers in the assembly and passes mount point information to the vServers through the vmapi.

Connection information for the ZFS storage device is configured using the existing target configuration interfaces of the Deployer. There is a new target type for this named “zfs”. This target type has the following settings:

- zfs.host
- zfs.port
- zfs.user
- zfs.pwd
- zfs.pool

- zfs.project

The operation for registering an assembly archive has an argument to declare the storage target to use. Also, the virtualization target definitions have a property, *storageTargets*, which declares the valid storage targets for use with the virtualization target. This exists because some storage targets may not be visible to the vServers created in a given virtualization target. Hence, the visibility is declared by the CloudAdmin who defines all the target information.

5.4.7.2 Application Lifecycle Scripts

The assembly resources file may contain shell scripts that are executed from the Deployer at various lifecycle points of an assembly. The resources file may contain a `scripts.d` directory with a set of subdirectories containing user scripts to be run at specific lifecycle points of the vServers started by the deployer. These scripts are executed from the Deployer process and are intended for performing actions in the deployment environment and not generally used to interact with the VMs themselves.

- **pre-deploy.d** - runs before deployment begins
- **post-deploy.d** - runs after deployment completes
- **deployer-pre-vm-start.d** - runs before the start of each vServer
- **deployer-post-vm-start.d** - runs after the start of each vServer
- **deployer-pre-app-config.d** - runs after the VM is up, and the network configured, but before the application on the VM is configured
- **deployer-post-app-config.d** - runs after the application on the VM is configured
- **pre-undeploy.d** - runs before undeployment begins
- **post-undeploy.d** - runs after undeployment is complete
- **deployer-pre-vm-stop.d** - runs before the stop of each vServer
- **deployer-post-vm-stop.d** - runs after the stop of each vServer

The scripts are passed a single argument which is a file path. This file path points to a properties file (name/value pairs) having the same set of properties that the Deployer

would pass to the VM over the VMAPI for that point in the lifecycle. The scripts are tied to lifecycle and not to any particular appliance or vserver. Scripts use the properties file passed in to infer any necessary context. If multiple scripts are included in a particular sub-directory, they are executed in alphabetical order.

If any script returns a non-zero result, it is considered failed and no more scripts would be run. The Deployer considers the transition failed at that point. One use case for the scripts is to configure environmental things external to the assembly itself, for example, modifying load balancer entries to allow a route to vservers created during deployment. Another possible use case is to send notifications of lifecycle events. Depending on the environment, it may or may not be possible to interact with the vServers themselves as there is no requirement that the Oracle Virtual Assembly Builder Deployer and the vServers be visible to one another on the same network.

For security reasons, if the assembly resource file contains scripts it can only be uploaded by a CloudAdmin.

5.4.7.3 Uploading Resources Using abctl

To upload assembly resources, use the `uploadAssemblyResources` command in `abctl`. Example 5–26 shows the uploadAssemblyResources command.

Example 5–26 uploadAssemblyArchive Command

```
$ ./abctl help -command uploadAssemblyArchive
$ ./abctl abctl uploadAssemblyResources -assemblyName myAssembly -version 1
-fileName resources.zip -connectionName myConnection
```

For more information see Appendix A, "Command Line Reference", which contains the details of the parameters that can be passed into the command.

5.4.8 Downloading an Assembly Archive from the Deployer Repository

You can download the assembly archive corresponding to an assembly version from the Deployer repository into the Oracle Virtual Assembly Builder Studio catalog. This may be necessary if you do not have a local copy of the assembly and you want to modify it or create a deployment plan.

5.4.8.1 Download Using Oracle Virtual Assembly Builder

To download, right-click on an assembly archive in the Deployments view, and select **Download Assembly Archive**.

5.4.8.2 Download Using abctl

Use the `downloadAssemblyArchive` command to unregister templates for an assembly. Example 5–27 shows the `downloadAssemblyArchive` command:

Example 5–27 downloadAssemblyArchive Command

```
$ ./abctl help -command downloadAssemblyArchive
$ ./abctl abctl downloadAssemblyArchive -name MyAssembly -version 1
RenamedAssembly.ova
```

For more information see Appendix A, "Command Line Reference", which contains the details of the parameters that can be passed into the command.

5.4.9 Registering an Assembly Archive to a Target

Once you upload an assembly archive to Oracle Virtual Assembly Builder Deployer you can register the assembly archive to a particular target.

5.4.9.1 Register Using Oracle Virtual Assembly Builder

Perform this in one of two ways:

- Open the *Deployments* navigator by selecting **View** > **Deployments**. Under Available Assemblies, right-click on a specific assembly archive version and select **Register**.
- Drag and drop a specific assembly archive version to the intended target. In the *Deployments* navigator, select the assembly archive under Available Assemblies, and drop it onto the target in Deployment Targets.

Both methods start the registration. The target node is updated with a new child representing the assembly archive version. The target node displays a throbber icon which will switch to the standard assembly archive icon once the registration is complete. You receive message feedback in the Message Log window as the registration progresses. If the registration fails, the node disappears and a pop-up menu appears explaining the failure.

To unregister, right-click on a registered node and select **Unregister**. The node will be removed once the unregistration is complete.

5.4.9.2 Register Using abctl

Use the `registerAssemblyArchive` command to unregister templates for an assembly. Example 5–28 shows the `unregisterAssemblyArchive` command:

Example 5–28 unregisterAssemblyArchive Command

```
$ ./abctl help -command registerAssemblyArchive
$ ./abctl registerAssemblyArchive -connectionName MyDeployerConnection -name
TheAssembly -version 1
```

For more information see Appendix A, "Command Line Reference", which contains the details of the parameters that can be passed into the command.

5.4.10 Deploying an Assembly Instance

This section describes how to deploy an assembly instance, using Oracle Virtual Assembly Builder Studio, or `abctl`.

When you deploy a registered assembly, an assembly instance and appliance instances are created and started. In Oracle Virtual Assembly Builder Studio, creation of the assembly instance, and deployment of the assembly instance are grouped into the "deploy" operation. In the abctl interface, you perform two separate commands.

Applications within the appliance instances are also started. Deploying an assembly instance can be a long running operation that can take several minutes. The time taken to deploy an assembly instance will vary depending upon the number of VMs that need to be created and started.

5.4.10.1 Deploy Using Oracle Virtual Assembly Builder Studio

Once an assembly archive is registered to a specific target, you can deploy that assembly archive one or more times.

1. Open the *Deployments* navigator by selecting **View** > **Deployments**.

   In the *Deployments* navigator, under the Deployment Targets pane, you can expand the targets and see a list of assembly archives that you can deploy.

2. Right-click an assembly version (assembly name and version) and select **Deploy**.

   You can also initiate an assembly instance by dragging a deployment plan from the Deployment Plan navigator and dropping it on a registered assembly archive.

3. Enter the following information in the *Deploy Assembly Archive* window.

   - **Deployment Plan**: select a deployment plan. The dialog populates the drop-down list with deployment plans matching the name of the assembly being deployed. If no plans exist, you must create a plan before you can deploy.
   - Vnet Mappings: verify (and possibly correct) the Vnet mappings in the deployment plan. At the time the deployment plan is created Oracle Virtual Assembly Builder cannot determine where you intend to deploy the assembly so it cannot validate the name of the target Vnet.

     You see one row for each Vnet defined in your assembly. The logical name of the Vnet (as defined in the assembly) is shown as the label and the corresponding drop-down contains the Vnets defined by the deployment

target. The value from the deployment plan is included in the drop-down list, even though it may be invalid, for the purposes of displaying to you the value from the deployment plan. The value, or mapping, from the deployment plan has an asterisk (*) suffix.

If the current mapping is invalid, the value displays a red error border and the "Deploy" button becomes disabled. You cannot deploy an assembly if the deployment plan contains invalid mappings. Any changes made to the Vnet mapping are saved to the deployment plan before the assembly instance initiates.

4. Click **Deploy** to update the deployment plan with the new mappings and initiate the assembly instance.

   Once you confirm the deployment options, the assembly deployment is initiated. A new child node is created under the registered node to indicate the progress. The name of the new child node is the Deployment ID. A throbber icon is shown until the assembly instance reaches a final state.

   When the network is initialized, you can expand an appliance to see the IP addresses of each virtual machine started for that appliance.

   When an appliance instance (and the assembly instance) is actively being processed you can see the throbber icon. As the assembly is deploying, you may see the icon for the appliance instances and possibly the assembly instance change from the throbber icon to a grey icon. The grey icon means the appliance instance has reached, and is currently sitting at, one the intermediate deployment states and it may stay in that state for several minutes while other appliance instances are being actively processed. This is especially noticeable when there are dependencies specified between appliances.

   When the appliance instance begins processing again, the icon returns to the throbber icon.

   If you select the assembly instance or appliance instance while the assembly is being deployed, you can see the current state in the property inspector.

   You can see the progress of the deployment in the Tasks navigator. If deployment fails, a red X mark is displayed in the navigator. You can open the task log for the failed task from the Tasks navigator using the **View Task Log** toolbar button.

5.4.10.2 Deploy Using abctl

Create an assembly instance by using the `createAssemblyInstance` command, as shown in Example 5–29. The `createAssemblyInstance` command will return the `assemblyInstanceId` needed for the `deployAssemblyInstance` command.

Example 5–29 createAssemblyInstance Command

```
$ ./abctl help -command createAssemblyInstance
$ ./abctl createAssemblyInstance -deploymentPlan c:/zeroAppliancesSite_plan.xml
-name SMALLOVA -version 1 -c cloudAdmin
Plan upload File Size: 700
Assembly Instance Id: gdc4_29x5_SMALLOVA_1
Assembly instance has been created.
```

Once you have created the assembly instance, you can deploy an assembly instance by using the `deployAssemblyInstance` command as shown in Example 5–30.

Example 5–30 deployAssemblyInstance Command (without Waiting for Completion)

```
$ ./abctl help -command deployAssemblyInstance
$ ./abctl deployAssemblyInstance -assemblyInstanceId gdc4_29x5_SMALLOVA_1 -c
cloudAdmin
Request ID: 1d1599a0-434b-426a-ab29-7c6230b5fa33
Request to deploy assembly instance has been submitted to deployer.
```

The `deployAssemblyInstance` command is an asynchronous operation. It will initiate a deploy request and return a request ID. The status of the request can be queried using the `describeRequests` operation.

If you want to wait until the completion of the operation, you may do so, optionally, by specifying additional parameters as follows:`-waitForComplete -pollTime 30`

Example 5–31 deployAssemblyInstance Command (Waiting for Completion)

```
$ ./abctl help -command deployAssemblyInstance
$ ./abctl deployAssemblyInstance -assemblyInstanceId gdc4_29x5_SMALLOVA_1 -c
cloudAdmin -waitForComplete -pollTime 30
Request ID: 1d1599a0-434b-426a-ab29-7c6230b5fa33
Request to deploy assembly instance has been submitted to deployer.
```

You can list the current deployments with the `describeAssemblyInstances` command, as shown in Example 5–32:

Example 5–32 describeAssemblyInstances Command

```
$ ./abctl help -command describeAssemblyInstances
$ ./abctl describeAssemblyInstances -c cloudAdmin
--------------------------------------------------------------------------------
 Name     | Version | State      | Assembly Instance Id | Target  | Appliances
--------------------------------------------------------------------------------
 SMALLOVA | 3       | Undeployed | c1Lm-GyML_SMALLOVA_3 | LOCBOX1 | c1Lm-GyML_
SMALLOVA_3:zeroAppliancesSite/myWls/Server_3
Assembly Instances have been described.
$
```

For more information see Appendix A, "Command Line Reference", which contains the details of the parameters that can be passed into the command.

5.4.11 Performing One-Step Deployment

One-step deployment consolidates several steps involved in creating and deploying an assembly archive into a single logical operation. Each of the steps is a potentially long running process and the consolidation eliminates the need for monitoring each individual step, waiting for it to complete and then starting the next. Instead, you can start the process to run over-night or over a weekend.

5.4.11.1 Prerequisites

The following conditions must be met before starting the process.

- All changes to the assembly must have been saved.
- All outputs must be connected.
- All file sets must have been captured.
- A deployer connection must exist and have at least one deployment target defined.

- If the assembly has an existing assembly archive, the platform type of the assembly archive must match the target platform:
  - OVM archives must be deployed to an OVM target.
- If the assembly does not yet have an assembly archive, a default base image must exist for the target platform, which is determined by the deployment target.
  - OVM targets require an OVM base image.
- At least one deployment plan must exist for the assembly and it must match the target platform.
  - OVM targets require an OVM deployment plan.
  - Template passwords must be set in the deployment plan.

If one of the prerequisites is not met, an error dialog describes why the assembly could not be deployed.

5.4.11.2 Deployment Considerations

The following conditions must be met addressed during deployment.

- You must specify the platform type. One-step deployment only shows deployment plans specific to the deployer connection type.
- By default, the deployment operation for an OVM target is an OVM archive.

One-step deployment consolidates the following steps:

1. Create Appliance Templates: Each appliance included in the assembly must have an appliance template for the target platform. If an appliance template already exists, it is used. Templates are only created for those appliances that do not already have a template for the target platform. If all appliances have existing templates, this step is skipped
2. Create Assembly Archive: The assembly must have an assembly archive for the target platform. If one does not already exist, one is created. If an archive needs to be created, it is compressed.

   If an assembly archive already exists for the target platform, this step is skipped.
3. Upload Assembly Archive: Once an assembly archive has been created, the archive is uploaded using the Deployer connection that was active (selected in the Deployments navigator) at the time the deployment was initiated. The description assigned to the uploaded assembly version is “one step deployment” and the version number is assigned automatically by the Deployer.
4. Register Assembly: Once the upload completes successfully, the newly uploaded assembly version is registered with the deployment target selected in the Deploy Assembly dialog.
5. Deploy Assembly: Once the registration completes successfully, the registered assembly is deployed using the deployment plan selected in the Deploy Assembly dialog.

5.4.11.3 Submitting an Assembly for Deployment

Once you have met the prerequisites, submit an assembly for one-step deployment, by selecting **Deploy** from the assembly context menu. Specify the platform type.

5.4.11.4 Task Tracking

After submitting an assembly for deployment, an entry for each step will appear in the Task Viewer. The first step, which may vary depending on the state of the assembly, begins execution immediately. The remaining steps are in a pending state until the preceding task completes successfully.

If any task fails, any pending tasks are cancelled and the process terminates.

5.4.12 Stopping an Assembly Instance

This section describes how to stop an assembly instance, using Oracle Virtual Assembly Builder Studio, or `abctl`.

When an assembly instance is stopped, the VMs and the applications that are running within the VMs are stopped. VMs that are in a stopped state retain their context. Stopped VMs can be restarted much more quickly than the original deployment because the VMs do not need to be created.

5.4.12.1 Stop an Assembly Instance with Oracle Virtual Assembly Builder Studio

In the Deployment Targets pane of the *Deployments* navigator you can start, stop, deploy, or undeploy an assembly instance. To stop an assembly instance, select the assembly instance that needs to be stopped and click **Stop**.

5.4.12.2 Stop an Assembly Instance with abctl

Use the `stopAssemblyInstance` command to stop an assembly instance. The assembly instance is referred to by its `assemblyInstanceId`. You can retrieve a list of assembly instances by using the `describeAssemblyInstances` command. Example 5–33 and Example 5–34 show the `stopAssemblyInstance` command:

Example 5–33 stopAssemblyInstance Command (without Waiting for Completion)

```
$ ./abctl help -command stopAssemblyInstance
$ ./abctl stopAssemblyInstance -assemblyInstanceId gdc4_29x5_SMALLOVA_1 -c
cloudAdmin
Request ID: 8486522b-8a5e-4348-bdf1-a7d55fccf848
Request for stop has been submitted to deployer.
```

Example 5–34 stopAssemblyInstance Command (Waiting for Completion)

```
$ ./abctl help -command stopAssemblyInstance
$ ./abctl stopAssemblyInstance -assemblyInstanceId gdc4_29x5_SMALLOVA_1 -c
cloudAdmin -waitForComplete -pollTime 30
Request ID: 8486522b-8a5e-4348-bdf1-a7d55fccf848
Request for stop has been submitted to deployer.
```

For more information see Appendix A, "Command Line Reference", which contains the details of the parameters that can be passed into the command.

5.4.13 Starting or Restarting an Assembly Instance

This section describes how to start or restart an assembly instance, using Oracle Virtual Assembly Builder Studio, or `abctl`. An assembly instance that has been stopped can be restarted. Restarting an assembly instance starts up all the VMs that were stopped and also starts up the applications within the VMs. The assembly instance gets restored to the state it was in before it was stopped. This operation completes more quickly than a deployment operation.

5.4.13.1 Restarting an Assembly Instance with Oracle Virtual Assembly Builder Studio

In the Deployment Targets pane of the *Deployments* navigator you can start, stop, deploy, or undeploy an assembly instance. To start an assembly instance, select the assembly instance and click **Start**.

5.4.13.2 Starting or Restarting an Assembly Instance with abctl

The `startAssemblyInstance` command is used to start a deployment. The assembly instance is referred to by its `assemblyInstanceId`. You can retrieve the list of deployments by using the `describeAssemblyInstances` command. Example 5–35 and Example 5–36 show the `startAssemblyInstance` command:

Example 5–35 Start an Assembly Instance (without Waiting for Completion)

```
$ ./abctl help -command startAssemblyInstance
$ ./abctl startAssemblyInstance -assemblyInstanceId gdc4_29x5_SMALLOVA_1 -c
cloudAdmin
Request ID: 1936dff2-f8a7-4407-83f8-08521bb48fef
Request for start has been submitted to deployer.
```

Example 5–36 Start an Assembly Instance (Waiting for Completion)

```
$ ./abctl help -command startAssemblyInstance
$ ./abctl startAssemblyInstance -assemblyInstanceId gdc4_29x5_SMALLOVA_1 -c
cloudAdmin -waitForComplete -pollTime 30
Request ID: 1936dff2-f8a7-4407-83f8-08521bb48fef
Request for start has been submitted to deployer.
```

The `restartAssemblyInstance` command is used to restart an assembly instance., and is equivalent to performing a `stopAssemblyInstance` followed by a `startAssemblyInstance`. The assembly instance is referred to by its `assemblyInstanceId`. You can retrieve the list of assembly instances by using the `describeAssemblyInstances` command. Example 5–37 and Example 5–38 show the `restartAssemblyInstance` command:

Example 5–37 Restarting an Assembly Instance (without Waiting for Completion)

```
$ ./abctl help -command restartAssemblyInstance
$ ./abctl restartAssemblyInstance -assemblyInstanceId gdc4_29x5_SMALLOVA_1 -c
cloudAdmin
Request ID: 126a97ef-89db-4b05-88d5-17b70e5cc3d2
Request to restart assembly instance has been submitted to deployer.
```

Example 5–38 Restarting an Assembly Instance (Waiting for Completion)

```
$ ./abctl help -command restartAssemblyInstance
$ ./abctl restartAssemblyInstance -assemblyInstanceId gdc4_29x5_SMALLOVA_1 -c
cloudAdmin -waitForComplete -pollTime 30
Request ID: 126a97ef-89db-4b05-88d5-17b70e5cc3d2
Request to restart assembly instance has been submitted to deployer.
```

For more information see Appendix A, "Command Line Reference", which contains the details of the parameters that can be passed into the command.

5.4.14 Scale Appliance

This section describes how to scale the selected appliance instance after an initial deployment of an assembly, using Oracle Virtual Assembly Builder Studio, or `abctl`.

After you deploy an assembly, the target number of VM instances for each appliance is started. The initial target for each appliance instance is specified in the deployment plan. You can dynamically specify a new target after an assembly has been deployed. Oracle Virtual Assembly Builder dynamically starts or stops VM instances to reach the new target (thus scaling up or scaling down). A scale down operation will only stop the properly deployed instances.

5.4.14.1 Scale Appliance with Oracle Virtual Assembly Builder Studio

To scale the number of instances of an appliance:

1. Select an assembly instance in the *Deployments* navigator and expand the assembly structure in the Structure Pane.
2. Right-click on a scalable appliance and select **Scale appliance**.
3. Set the target number of VM instances for the appliance from the *Target Scale* drop down. The current Scale level is identified by a * next to the right value. For example: 1* or 2*.
4. Click **OK**.

5.4.14.2 Retrieve the scalingGroupId for Use in the Scale Command

Use the `describeScalingGroups` command to retrieve the `scalingGroupId` to pass in to the `scale` command. (Example 5–39):

Example 5–39 describeScalingGroup Command

```
$ ./abctl help -command describeScalingGroups
$ ./abctl describeScalingGroups
```

For more information see Appendix A, "Command Line Reference", which contains the details of the parameters that can be passed into the command.

5.4.14.3 Scale Appliance(s) in an Assembly Instance with abctl

Use the `scale` command to scale the appliance (Example 5–40 and Example 5–41):

Example 5–40 scale Command (without Waiting for Completion)

```
$ ./abctl help -command scale
$ ./abctl scale -scalingGroupId 1gWT-t0Np_SMALLOVA_
1:zeroAppliancesSite/myWls/Server_3 -target 1 -c cloudAdmin
Request ID: c1d2c742-d2fe-4698-bf61-99d619be4fca
Request for scaling operation has been submitted to deployer.
```

Example 5–41 scale Command (Waiting for Completion)

```
$ ./abctl help -command scale
$ ./abctl scale -scalingGroupId 1gWT-t0Np_SMALLOVA_
1:zeroAppliancesSite/myWls/Server_3 -target 1 -c cloudAdmin -waitForComplete
-pollTime 30
Request ID: c1d2c742-d2fe-4698-bf61-99d619be4fca
Request for scaling operation has been submitted to deployer.
```

For more information see Appendix A, "Command Line Reference", which contains the details of the parameters that can be passed into the command.

5.4.15 Undeploying an Assembly Instance

This section describes how to undeploy an assembly instance, using Oracle Virtual Assembly Builder Studio, or `abctl`.

Undeploying an assembly instance stops all the running VMs. It also cleans up any failed VMs that may exist.

5.4.15.1 Undeploying a Deployment with Oracle Virtual Assembly Builder Studio

In the *Deployments* navigator, you can undeploy an assembly instance by right-clicking on a deployed assembly instance and selecting **Undeploy**. If successful, the node is removed. If the operation fails, a red icon appears. Progress messages are shown in the Message Log window.

5.4.15.2 Undeploying an Assembly Instance with abctl

You can use the `undeployAssemblyInstance` command to undeploy an assembly instance. The assembly instance is referred to by its `assemblyInstanceId`. You can retrieve a list of assembly instances by using the `describeAssemblyInstances` command. Example 5–42 shows the `undeployAssemblyInstance` command:

Example 5–42 undeployAssemblyInstance Command (without Waiting for Completion)

```
$ ./abctl help -command undeployAssemblyInstance
$ ./abctl undeployAssemblyInstance -assemblyInstanceId gdc4_29x5_SMALLOVA_1 -c
cloudAdmin
Request ID: 6b0f1b14-466e-4b23-bcc3-8b8506fd40ac
Request to undeploy assembly instance has been submitted to deployer.
```

Example 5–43 undeployAssemblyInstance Command (Waiting for Completion)

```
$ ./abctl help -command undeployAssemblyInstance
$ ./abctl undeployAssemblyInstance -assemblyInstanceId gdc4_29x5_SMALLOVA_1 -c
cloudAdmin -waitForComplete -pollTime 30
Request ID: 6b0f1b14-466e-4b23-bcc3-8b8506fd40ac
Request to undeploy assembly instance has been submitted to deployer.
```

For more information see Appendix A, "Command Line Reference", which contains the details of the parameters that can be passed into the command.

5.4.16 Unregistering an Assembly Archive from a Target

This section describes how to unregister an assembly archive from Oracle Virtual Assembly Builder Deployer using `abctl`. This operation unregisters the assembly archive, but does not delete the uploaded archive.

5.4.16.1 Unregistering Assembly Archives with Oracle Virtual Assembly Builder Studio

Unregister an assembly archive version (from a deployment target) by right-clicking on a registered version and selecting **Unregister**.

5.4.16.2 Unregistering Assembly Archives with abctl

Use the `unregisterAssemblyArchive` command to unregister templates for an assembly. Example 5–44 and Example 5–45 show the `unregisterAssemblyArchive` command:

Example 5–44 unregisterAssemblyArchive Command (without Waiting for Completion)

```
$ ./abctl help -command unregisterAssemblyArchive
$ ./abctl unregisterAssemblyArchive -name SMALLOVA -version 1 -target LOCBOX1 -c
cloudAdmin
Request ID: f9f9d0b7-e334-4020-a038-2b728e9a0a37
Request to unregister assembly has been submitted to deployer.
```

Example 5–45 unregisterAssemblyArchive Command (Waiting for Completion)

```
$ ./abctl help -command unregisterAssemblyArchive
$ ./abctl unregisterAssemblyArchive -name SMALLOVA -version 1 -target LOCBOX1 -c
cloudAdmin -waitForComplete -pollTime 30
Request ID: f9f9d0b7-e334-4020-a038-2b728e9a0a37
Request to unregister assembly has been submitted to deployer.
```

For more information see Appendix A, "Command Line Reference", which contains the details of the parameters that can be passed into the command.

5.4.16.3 redeployAssemblyInstance

Use the `redeployAssemblyInstance` command to redeploy an assembly instance. Example 5–46 and Example 5–46 show the `redeployAssemblyInstance` command:

Example 5–46 redeployAssemblyInstance Command (without Waiting for Completion)

```
$ ./abctl help -command redeployAssemblyInstance
$ ./abctl redeployAssemblyInstance -assemblyInstanceId gdc4_29x5_SMALLOVA_1 -c
cloudAdmin
Request ID: eff86a4c-d064-4794-b1ae-0624a972ab06
Request to redeploy assembly instance has been submitted to deployer.
```

Example 5–47 redeployAssemblyInstance Command (Waiting for Completion)

```
$ ./abctl help -command redeployAssemblyInstance
$ ./abctl redeployAssemblyInstance -assemblyInstanceId gdc4_29x5_SMALLOVA_1 -c
cloudAdmin -waitForComplete -pollTime 30
Request ID: eff86a4c-d064-4794-b1ae-0624a972ab06
Request to redeploy assembly instance has been submitted to deployer.
```

For more information see Appendix A, "Command Line Reference", which contains the details of the parameters that can be passed into the command.

5.4.16.4 deleteAssemblyInstance

Use the `deleteAssemblyInstance` command to delete an assembly instance once the assembly instance is in an undeployed state. Example 5–48 shows the `deleteAssemblyInstance` command:

Example 5–48 deleteAssemblyInstance Command

```
$ ./abctl help -command deleteAssemblyInstance
$ ./abctl deleteAssemblyInstance -assemblyInstanceId gdc4_29x5_SMALLOVA_1 -c
cloudAdmin
Assembly instance gdc4_29x5_SMALLOVA_1 has been deleted.
```

For more information see Appendix A, "Command Line Reference", which contains the details of the parameters that can be passed into the command.

5.4.17 Deleting an Assembly Archive from the Deployer Repository

This section describes how to delete an assembly archive from the Deployer repository, using Oracle Virtual Assembly Builder Studio, or `abctl`.

5.4.17.1 Deleting an Assembly Archive using Oracle Virtual Assembly Builder Studio

Delete an assembly archive from the Deployer repository by right clicking on the uploaded archive and selecting **Delete Assembly Archive**.

5.4.17.2 Deleting an Assembly Archive using abctl

The OVAB Admin can use the `deleteAssemblyArchive` command in `abctl` to delete the assembly archive from the Deployer repository. This operation may only be performed if there are no registrations for the assembly archive.

Example 5–49 deleteAssemblyArchive Command (without Waiting for Completion)

```
$ ./abctl help -command deleteAssemblyArchive
$ ./abctl deleteAssemblyArchive -name TheAssembly -version 1 -connectionName
myDeployerConnection
```

Example 5–50 deleteAssemblyArchive Command (Waiting for Completion)

```
$ ./abctl help -command deleteAssemblyArchive
$ ./abctl deleteAssemblyArchive -name TheAssembly -version 1 -connectionName
myDeployerConnection -waitForComplete -pollTime 30
```

For more information see Appendix A, "Command Line Reference", which contains the details of the parameters that can be passed into the command.

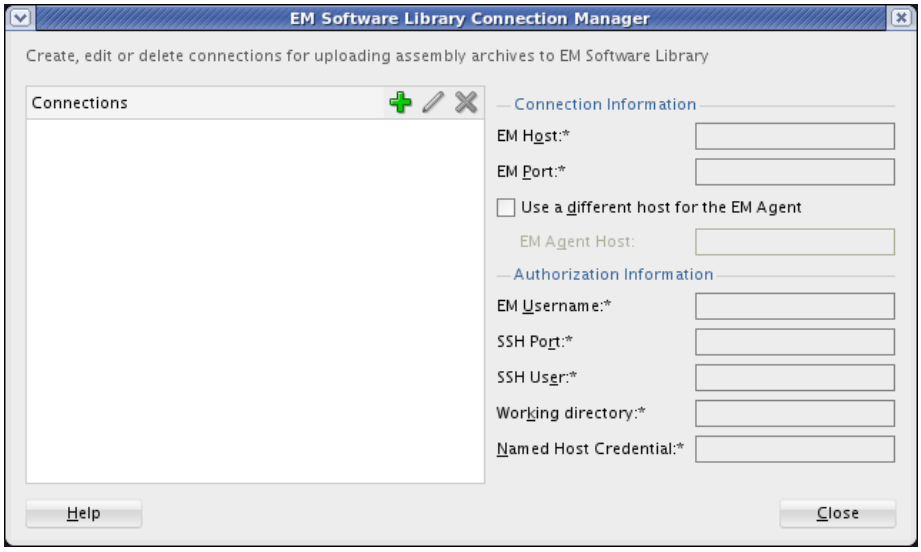
5.4.18 Interacting with EM Software Library

Use these operations if you plan to use Enterprise Manager Grid Control Cloud Management Pack to perform deployment operations on assembly archives. In this scenario, you use Oracle Virtual Assembly Builder to create your assembly archive and upload it to EM Software Library. You use Enterprise Manager Grid Control Cloud Management Pack to deploy the assembly instance.

5.4.18.1 Managing Connection to EM Software Library with Oracle Virtual Assembly Builder Studio

You can manage connections to the EM Software Library by selecting **Tools** > **Manage EM Software Library Connection**. The EM Software Libary Connection Manager dialog appears (Figure 5–13), which allows you to add, edit, or delete connections for uploading assembly archives to EM Software Library. You can create only one connection to EM Software Library.

Figure 5–13 EM Software Library Connection Manager



This graphic displays the EM Software Library Connection Manager, which is described in the surrounding text.

****

To create the connection to EM Software Library:

1. Select the create icon.
2. Configure the following connection properties:
   - EM Host: enter a host for the connection.
   - Port: enter a port for the connection.
   - EM Username: enter the username for accessing Enterprise Manager.
   - SSH Port: Enter the SSH port for the connection.
   - SSH User: Enter the SSH user for the connection.
   - SSH Authentication: Select **Password** and enter a password or select **Private** Key to reference the SSH key to use rather than providing a password. If selecting Private Key, select the browse button and navigate to the location of a private SSH key file on the local machine. The use of a private key file provides added security because no password handling is required by Oracle Virtual Assembly Builder.
   - Working directory: Enter the working directory where assembly archives are uploaded to Oracle Software Library.
   - Named host credential: Enter the named host credential that specifies authentication information for Oracle Software Library.
3. Click **Test Connection** to verify that you can successfully connect.
4. Click **OK**.

To edit the connection to EM Software Library:

1. Select the connection in the Connections pane.
2. Select the edit icon.
3. Updates the connection properties as required.

4. Click **OK**.

To delete the connection to EM Software Library:

1. Select the connection in the Connections pane.
2. Select the delete icon.
3. Confirm the deletion.

5.4.18.2 Creating a Connection to Oracle Software Library Using abctl

You can configure a connection to the Oracle Software Library using the `createEMConnection` command in `abctl`. The connection to Oracle Software Library is persisted in a connections file.

You must specify the fully qualified hostname of the remote Enterprise Manager machine, for example `myhost.example.com` instead of `myhost`.

Example 5–51 createEMConnection

```
$ ./abctl createEmConnection -connectionURL emMachine:7791 -connectionUser admin
-namedHostCredential hostCredential -remoteUser mySshUser -remoteWorkingDir
/scratch/myovas [-sshPort 23] [-privateKeyFile ~/.ssh/id_rsa]
```

You are prompted for a `connectionPassword`.

For more information see Appendix A, "Command Line Reference", which contains the details of the parameters that can be passed into the command.

5.4.18.3 Uploading an Assembly

You can upload an assembly archive from Oracle Virtual Assembly Builder Studio Catalog to Oracle Software Library using Oracle Virtual Assembly builder Studio or `abctl`.

To upload an assembly archive in Oracle Virtual Assembly Builder Studio, right click the assembly and select **Upload Assembly Archive** > **EM Software Library**.

To upload an assembly archive in `abctl`, use the `uploadEMAssemblyArchive` command in `abctl`. The assembly archive must have been created with the assembly.

Example 5–52 Upload an Assembly to Oracle Software Library

```
$ ./abctl uploadEMAssemblyArchive -name archiveName -description "my assembly
archive"
 Assembly archive upload started
  Assembly archive upload at 10%
  Assembly archive upload at 20%
  Assembly archive upload at 30%
  Assembly archive upload at 40%
  Assembly archive upload at 50%
  Assembly archive upload at 60%
  Assembly archive upload at 70%
  Assembly archive upload at 80%
  Assembly archive upload at 90%
  Assembly archive upload at 100%
  Assembly archive upload complete
  Assembly archive version 0.1 uploaded
  Successfully uploaded the assembly archive mySite to EM Software Library.  Check
the status of the assembly archive with describeEMAssemblyArchives before using.
```

For more information see Appendix A, "Command Line Reference", which contains the details of the parameters that can be passed into the command.

5.4.18.4 Describe Assembly Archives in Oracle Software Library

You can list the assembly archives and their versions present in Oracle Software Library using the `describeEMAssemblyArchives` command in `abctl`:

Example 5–53 Describing the Assembly Archives in EM Software Library

```
$ ./abctl describeEMAssemblyArchives [-name nameOfAssemblyArchive]

-----------------------------------------------
Name   | Version | Description | Status
-----------------------------------------------
mySite | 0.3     | mysite3     | READY
       | 0.2     | mysite3     | READY
       | 0.1     | mysite3     | READY
-----------------------------------------------
```

For more information see Appendix A, "Command Line Reference", which contains the details of the parameters that can be passed into the command.

5.4.18.5 Deleting an Assembly Archive from Oracle Software Library

You can delete the specified version of an assembly archive from Oracle Software Library using the `deleteEMAssemblyArchive` command in `abctl`.

> **Note:** Do not attempt to delete an assembly archive until its status is READY from the `describeEMAssemblyArchives` command. See Example 5–53, "Describing the Assembly Archives in EM Software Library" for example output.

Example 5–54 Delete an Assembly Archive from Oracle Software Library

```
abctl deleteEMAssemblyArchive -name archiveName -version 1.2
```

For more information see Appendix A, "Command Line Reference", which contains the details of the parameters that can be passed into the command.

5.4.18.6 Downloading an Assembly to Oracle Virtual Assembly Builder Catalog

You can download an assembly from the EM Software Library to the Oracle Virtual Assembly Builder Studio Catalog using Oracle Virtual Assembly Builder Studio or `abctl`.

To download an assembly using Oracle Virtual Assembly Builder Studio, right click the assembly in the EM Software Library view and select **Download Assembly Archive**.

> **Note:** Do not attempt to download an assembly archive until its status is READY from the `describeEMAssemblyArchives` command. See Example 5–53, "Describing the Assembly Archives in EM Software Library" for example output.

The assembly archive is reverse engineered to have the Oracle Virtual Assembly Builder metadata, file sets and templates created and persisted in the Oracle Virtual Assembly Builder Studio catalog.

To download an assembly from Oracle Software Library to the Oracle Virtual Assembly Builder Studio Catalog, use the `download` command in `abctl`.

Example 5–55 Download an Assembly

```
abctl downloadEMAssemblyArchive -name archiveName -version 1.0 -force -downloadAs
newName
```

For more information see Appendix A, "Command Line Reference", which contains the details of the parameters that can be passed into the command.

A

Command Line Reference

This appendix contains information about the `abctl` commands included in Oracle Virtual Assembly Builder.

A.1 Commands

This section describes the available commands. Commands fall into two categories:

- Commands for creating appliances and assemblies, creating appliance templates and assembly archives, and creating deployment plans.
- Commands for configuring deployment targets, uploading assembly archives to Deployer, creating assembly instances, and deploying, undeploying, starting, and stopping assembly instances and scaling appliance instances. These commands interface with the Oracle Virtual Assembly Builder Deployer Web service.

When Oracle Virtual Assembly Builder Studio is installed, all commands are available. When only Oracle Virtual Assembly Builder Deployer is installed, a subset of the commands are available. Table A–1 indicates which commands are available depending on which Oracle Virtual Assembly Builder components have been installed.

Table A–1 Available Commands by Installation Type

Command Name and Section	Command Availability for Installation Type		
	Installation of Deployer only	Installation of Studio only	Installation of Studio and Deployer
Section A.1.1, "addAssemblyUsers"	Yes	Yes	Yes
Section A.1.2, "addTargetUser"	Yes	Yes	Yes
Section A.1.3, "addToAssembly"	No	Yes	Yes
Section A.1.4, "bindAssemblyFileSetDefinition"	No	Yes	Yes
Section A.1.5, "bindInput"	No	Yes	Yes
Section A.1.6, "bindInterface"	No	Yes	Yes
Section A.1.7, "captureFileSets"	No	Yes	Yes
Section A.1.8, "clearAssemblyPasswords"	No	Yes	Yes
Section A.1.9, "connectEndpoints"	No	Yes	Yes

Table A–1 (Cont.) Available Commands by Installation Type

Command Name and Section	Command Availability for Installation Type		
	Installation of Deployer only	Installation of Studio only	Installation of Studio and Deployer
Section A.1.10, "copy"	No	Yes	Yes
Section A.1.11, "copyAssemblyArchive"	No	Yes	Yes
Section A.1.12, "createAssembly"	No	Yes	Yes
Section A.1.13, "createAssemblyArchive"	No	Yes	Yes
Section A.1.14, "createAssemblyFileSetDefinition"	No	Yes	Yes
Section A.1.15, "createAssemblyInstance"	Yes	Yes	Yes
Section A.1.16, "createDeployerConnection"	Yes	Yes	Yes
Section A.1.17, "createEMConnection"	No	Yes	Yes
Section A.1.18, "createExternalResources"	No	Yes	Yes
Section A.1.19, "createInterface"	No	Yes	Yes
Section A.1.20, "createNativeFileSetDefinition"	No	Yes	Yes
Section A.1.21, "createNFSFileSetDefinition"	No	Yes	Yes
Section A.1.22, "createRAWFileSetDefinition"	No	Yes	Yes
Section A.1.23, "createTags"	Yes	Yes	Yes
Section A.1.24, "createTarget"	Yes	Yes	Yes
Section A.1.25, "createTemplate"	No	Yes	Yes
Section A.1.26, "createVnet"	No	Yes	Yes
Section A.1.27, "delete"	No	Yes	Yes
Section A.1.28, "deleteAssemblyArchive"	Yes	Yes	Yes
Section A.1.29, "deleteAssemblyInstance"	Yes	Yes	Yes
Section A.1.30, "deleteAssemblyResources"	Yes	Yes	Yes
Section A.1.31, "deleteDeployerConnection"	Yes	Yes	Yes
Section A.1.32, "deleteDeploymentPlan"	Yes	Yes	Yes
Section A.1.33, "deleteEMAssemblyArchive"	No	Yes	Yes
Section A.1.34, "deleteEMConnection"	No	Yes	Yes
Section A.1.35, "deleteFailedApplianceInstances"	Yes	Yes	Yes
Section A.1.36, "deleteFileSetDefinition"	No	Yes	Yes
Section A.1.37, "deleteInterface"	No	Yes	Yes
Section A.1.38, "deleteLogEvents"	Yes	Yes	Yes
Section A.1.39, "deleteRequests"	Yes	Yes	Yes
Section A.1.40, "deleteTags"	Yes	Yes	Yes
Section A.1.41, "deleteTarget"	Yes	Yes	Yes
Section A.1.42, "deleteVnet"	No	Yes	Yes

Table A–1 (Cont.) Available Commands by Installation Type

Command Name and Section	Command Availability for Installation Type		
	Installation of Deployer only	Installation of Studio only	Installation of Studio and Deployer
Section A.1.43, "deployAssemblyInstance"	Yes	Yes	Yes
Section A.1.44, "describeApplianceInstanceMetrics"	Yes	Yes	Yes
Section A.1.45, "describeApplianceInstances"	Yes	Yes	Yes
Section A.1.47, "describeAssemblyArchives"	Yes	Yes	Yes
Section A.1.48, "describeAssemblyInstances"	Yes	Yes	Yes
Section A.1.49, "describeAssemblyResources"	Yes	Yes	Yes
Section A.1.50, "describeAssemblyUsers"	Yes	Yes	Yes
Section A.1.51, "describeAssemblyVnets"	No	Yes	Yes
Section A.1.52, "describeCatalog"	No	Yes	Yes
Section A.1.53, "describeDeployer"	Yes	Yes	Yes
Section A.1.54, "describeDeployerConnections"	Yes	Yes	Yes
Section A.1.55, "describeDeploymentPlans"	No	Yes	Yes
Section A.1.56, "describeEMAssemblyArchives"	No	Yes	Yes
Section A.1.57, "describeEMConnection"	No	Yes	Yes
Section A.1.58, "describeEndpoints"	No	Yes	Yes
Section A.1.59, "describeFileSetDefinitions"	No	Yes	Yes
Section A.1.60, "describeInterfaces"	No	Yes	Yes
Section A.1.61, "describeLogEvents"	No	Yes	Yes
Section A.1.62, "describePlugins"	No	Yes	Yes
Section A.1.63, "describeRegistrations"	Yes	Yes	Yes
Section A.1.64, "describeRequests"	Yes	Yes	Yes
Section A.1.65, "describeResources"	Yes	Yes	Yes
Section A.1.66, "describeScalingGroups"	Yes	Yes	Yes
Section A.1.67, "describeTags"	Yes	Yes	Yes
Section A.1.68, "describeTargetConfigurations"	Yes	Yes	Yes
Section A.1.69, "describeTargetNames"	Yes	Yes	Yes
Section A.1.70, "describeTargetUsers"	Yes	Yes	Yes
Section A.1.71, "describeTargets"	Yes	Yes	Yes
Section A.1.72, "describeUserTargets"	Yes	Yes	Yes
Section A.1.73, "describeVnets"	Yes	Yes	Yes
Section A.1.74, "disablePlugin"	No	Yes	Yes
Section A.1.75, "downloadAssemblyArchive"	Yes	Yes	Yes
Section A.1.76, "downloadAssemblyMetadata"	Yes	Yes	Yes
Section A.1.77, "downloadAssemblyResources"	Yes	Yes	Yes

Table A–1 (Cont.) Available Commands by Installation Type

Command Name and Section	Command Availability for Installation Type		
	Installation of Deployer only	Installation of Studio only	Installation of Studio and Deployer
Section A.1.78, "downloadDeploymentPlan"	Yes	Yes	Yes
Section A.1.79, "downloadEMAssemblyArchive"	No	Yes	Yes
Section A.1.80, "enablePlugin"	No	Yes	Yes
Section A.1.81, "encryptProperties"	No	Yes	Yes
Section A.1.82, "encryptProperty"	No	Yes	Yes
Section A.1.83, "export"	No	Yes	Yes
Section A.1.84, "findPlugins"	No	Yes	Yes
Section A.1.85, "getCatalogProperty"	No	Yes	Yes
Section A.1.86, "getDefaultTarget"	Yes	Yes	Yes
Section A.1.87, "getTargetType"	Yes	Yes	Yes
Section A.1.88, "help"	Yes	Yes	Yes
Section A.1.89, "import"	No	Yes	Yes
Section A.1.90, "installPlugins"	No	Yes	Yes
Section A.1.91, "introspectCoherenceWeb"	No	Yes	Yes
Section A.1.92, "introspectForms"	No	Yes	Yes
Section A.1.93, "introspectGenericProd"	No	Yes	Yes
Section A.1.94, "introspectOHS"	No	Yes	Yes
Section A.1.95, "introspectOTD"	No	Yes	Yes
Section A.1.96, "introspectRACDB"	No	Yes	Yes
Section A.1.97, "introspectReports"	No	Yes	Yes
Section A.1.98, "introspectSIDB"	No	Yes	Yes
Section A.1.99, "introspectSOA"	No	Yes	Yes
Section A.1.100, "introspectTuxedo"	No	Yes	Yes
Section A.1.101, "introspectWLS"	No	Yes	Yes
Section A.1.102, "redeployAssemblyInstance"	Yes	Yes	Yes
Section A.1.103, "registerAssemblyArchive"	Yes	Yes	Yes
Section A.1.104, "removeAssemblyUsers"	Yes	Yes	Yes
Section A.1.105, "removePlugin"	No	Yes	Yes
Section A.1.106, "removeTargetUsers"	Yes	Yes	Yes
Section A.1.107, "renameExternalResource"	No	Yes	Yes
Section A.1.108, "restartAssemblyInstance"	Yes	Yes	Yes
Section A.1.109, "resumeAssemblyInstance"	Yes	Yes	Yes
Section A.1.110, "scale"	Yes	Yes	Yes
Section A.1.111, "scaleDown"	Yes	Yes	Yes

Table A–1 (Cont.) Available Commands by Installation Type

Command Name and Section	Command Availability for Installation Type		
	Installation of Deployer only	Installation of Studio only	Installation of Studio and Deployer
Section A.1.112, "setCatalogProperty"	No	Yes	Yes
Section A.1.113, "setDefaultTarget"	Yes	Yes	Yes
Section A.1.114, "startAssemblyInstance"	Yes	Yes	Yes
Section A.1.115, "stopAssemblyInstance"	Yes	Yes	Yes
Section A.1.116, "suspendAssemblyInstance"	Yes	Yes	Yes
Section A.1.117, "unbindAssemblyFileSetDefinition"	No	Yes	Yes
Section A.1.118, "undeployAssemblyInstance"	Yes	Yes	Yes
Section A.1.119, "unregisterAssemblyArchive"	Yes	Yes	Yes
Section A.1.120, "updateAssemblyArchive"	Yes	Yes	Yes
Section A.1.121, "updateDeployerConfig"	Yes	Yes	Yes
Section A.1.122, "updateTarget"	Yes	Yes	Yes
Section A.1.123, "uploadAssemblyArchive"	Yes	Yes	Yes
Section A.1.124, "uploadAssemblyResources"	Yes	Yes	Yes
Section A.1.125, "uploadDeploymentPlan"	Yes	Yes	Yes
Section A.1.126, "uploadEMAssemblyArchive"	No	Yes	Yes
Section A.1.127, "validateAssemblyArchiveResources"	Yes	Yes	Yes
Section A.1.128, "validateAssemblyInstanceResources"	Yes	Yes	Yes
Section A.1.129, "verifyEMConnection"	No	Yes	Yes
Section A.1.130, "version"	Yes	Yes	Yes

A.1.1 addAssemblyUsers

Details for this command follow.

A.1.1.1 Synopsis

```
$ abctl addAssemblyUsers -assembly String -user String... -connectionName String
```

A.1.1.2 Description

Adds one or more users to an assembly.

A.1.1.3 Options

Table A–2 shows the command options for `addAssemblyUsers`.

Table A–2 addAssemblyUsers options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-assembly`	a	true	none	A string representing the name of the assembly.	Specifies the assembly to which to add users
`-connection Name`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
`-user`	u	true	none	A string representing the username(s) to add.	Specifies one or more users to add to an assembly.

A.1.1.4 Examples

Here are some command examples.

A.1.1.4.1 Adding Users to an Assembly

```
$ abctl addAssemblyUsers -assembly MyAssembly -user User1 User2
```

A.1.2 addTargetUser

Details for this command follow.

A.1.2.1 Synopsis

```
$ abctl addTargetUser -user String -target String [-properties String...]
-connectionName String
```

A.1.2.2 Description

Adds a user to a target.

A.1.2.3 Options

Table A–3 shows the command options for `addTargetUser`.

Table A–3 addTargetUser options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-user`	u	true	none	A string representing the username of the user.	The username of the user.
`-target`	t	true	none	A string representing the target to which the user is being added.	The target to which a user is being added.
`-properties`	p	false	none	A string representing property=value pairs to apply to the user.	The properties to apply to the user.
`-connection Name`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.2.4 Examples

Here are some command examples.

A.1.2.4.1 Adding Several Users to an Assembly

```
$ abctl addTargetUser -user Username -target Targetname -connectionName
MyDeployerConnection
```

A.1.3 addToAssembly

Details for this command follow.

A.1.3.1 Synopsis

```
$ abctl addToAssembly -name String -into String
```

A.1.3.2 Description

Adds an existing appliance or atomic assembly to another existing assembly.

Sharing is continued when you add an assembly to another assembly. That is, templates and file sets in the original assembly are shared with the new nested assembly.

A.1.3.3 Options

Table A–4 shows the command options for `addToAssembly`.

Table A–4 addToAssembly options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-name`	n	true	none	The name of an existing appliance or atomic assembly.	The name of an existing appliance or atomic assembly to add.
`-into`	i	true	none	The name of a non-atomic assembly.	The name of an existing assembly to populate.

A.1.3.4 Examples

Here are some command examples.

A.1.3.4.1 Adding an Appliance into an Existing Assembly

```
$ abctl addToAssembly -name myAppliance -into myAssembly
```

A.1.4 bindAssemblyFileSetDefinition

Details for this command follow.

A.1.4.1 Synopsis

```
$ abctl bindAssemblyFileSetDefinition -name String -to String
```

A.1.4.2 Description

Binds an assembly file set definition to an appliance. This appliance must be nested within the assembly containing that file set definition. Binding it to one appliance in an atomic assembly binds it to all appliances in that atomic assembly.

A.1.4.3 Options

Table A–5 shows the command options for `bindAssemblyFileSetDefinition`.

Table A–5 bindAssemblyFileSetDefinition options

Name	Alias	Req'd	Default Values	Possible Values	Description
-name	n	true	none	N/A	Name of assembly file set definition.
-to	t	true	none	N/A	Path to nested appliance.

A.1.4.4 Examples

Here are some command examples.

A.1.4.4.1 Bind Assembly File Set Definition

```
% abctl bindAssemblyFileSetDefinition -name middleware -to mySite/ohs1
```

A.1.5 bindInput

Details for this command follow.

A.1.5.1 Synopsis

```
$ abctl bindInput -name String -appliance String -interface String
```

A.1.5.2 Description

Binds an existing appliance input to one of the existing network interfaces of the appliance. Specify the interface name "INADDR_ANY" to cause the input to be bound to all available interfaces rather than any specific network interface.

A.1.5.3 Options

Table A–6 shows the command options for `bindInput`.

Table A–6 bindInput options

Name	Alias	Req'd	Default Values	Possible Values	Description
-name	n	true	none	N/A	The name of the input to bind.
-appliance	a	true	none	N/A	The name of an existing appliance.
-interface	i	true	none	N/A	The name of the interface to which to bind the input, or "INADDR_ANY".

A.1.5.4 Examples

Here are some command examples.

A.1.5.4.1 Bind Input

```
% abctl bindInput -name myInput -appliance mySite/myOhs -interface interface-1
```

A.1.6 bindInterface

Details for this command follow.

A.1.6.1 Synopsis

```
$ abctl bindInterface -name String -appliance String -vnet String
```

A.1.6.2 Description

Binds an existing appliance network interface to one of the existing Vnets of the containing assembly.

A.1.6.3 Options

Table A–7 shows the command options for `bindInterface`.

Table A–7 bindInterface options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-name`	n	true	none	N/A	The name of the interface to bind.
`-appliance`	a	true	none	N/A	The name of an existing appliance. The appliance must be contained in an assembly.
`-vnet`	v	true	none	N/A	The name of the network to which to bind the interface.

A.1.6.4 Examples

Here are some command examples.

A.1.6.4.1 Bind Interface

```
% abctl bindInterface -name myInterface -appliance mySite/myOhs -vnet vnet-1
```

A.1.7 captureFileSets

Details for this command follow.

A.1.7.1 Synopsis

```
$ abctl captureFileSets -name String [-remoteHost String] [-remoteUser String]
[-sudoUser String] [-remoteWorkingDir Path] [-remoteCleanup]
[-privateKeyFile Path] [-quiet] [-force]
```

A.1.7.2 Description

Creates file sets for the specified appliance or assembly.

A.1.7.3 Options

Table A–10 shows the command options for `captureFileSets`.

Table A–8 captureFileSets options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-remoteHost`	rh	false	none	N/A	Host name or IP address and optional SSH port of the remote machine. If set, the `remoteUser` must be specified as well.
`-remoteUser`	ru	false	none	N/A	Name of the ssh user to use for accessing the remote machine. If set, the `remoteHost` must be specified as well.
`-sudoUser`	su	false	none	User name of sudo user.	Name of the user on the remote machine to sudo as before executing operations. Note that sudoUser is equivalent to *Run as user* in Oracle Virtual Assembly Builder Studio. If sudoUser is specified, you cannot use the privateKeyFile. That is, sudoUser can only be used when you provide a password.
`-remoteWorkingDir`	rwd	false	`/tmp/abRemote_<remote user name>`	N/A	Path on the remote machine to work out of. If set, the `remoteUser` and `remoteHost` must be specified as well.
`-remoteCleanup`	rc	false	none	N/A	Remote clean up flag. When set, the remote working directory is deleted after the operation. Otherwise the directory is not be modified. If set, the `remoteUser` and `remoteHost` must be specified as well.
`-privateKeyFile`	pkf	false	~/.ssh/id_rsa	Location of a private key file.	Private SSH key file on the local machine.
`-quiet`	q	false	none	N/A	By default, the command shows detailed progress/success messages. If `-quiet` is set, the command turns off verbose mode and shows only one or two progress/success messages.
`-name`	n	true	Derived directory name prefixed by component type name.	Name of an appliance or assembly. Nested appliances or assemblies are referred to with slash ('/'), for example: mySite/myOhs	Specify the name of an appliance or assembly to be captured. For an assembly, only an atomic assembly name can be specified. To capture a non-atomic assembly, its sub-appliances and sub-assemblies must be captured individually.
`-force`	f	false	none	N/A	If specified, existing file sets and any appliance templates created from it will be overwritten. The operation can fail if there is an existing registered appliance template that was created from an existing file set. The flag has no effect if no file set or template exists.

A.1.7.4 Examples

Here are some command examples.

A.1.7.4.1 Capture File Sets

```
% abctl captureFileSets -name myOhs -force
```

A.1.8 clearAssemblyPasswords

Details for this command follow.

A.1.8.1 Synopsis

```
$ abctl clearAssemblyPasswords -name String
```

A.1.8.2 Description

Clears all passwords for a top-level non-atomic assembly and its child appliances and assemblies, such that all passwords must be provided within the deployment plan. Removes all user and template passwords (does not pertain to system property passwords since they are not reflected in the deployment plan). Optional passwords that have no value at the time the command is executed are still not required to be provided in the deployment plan. Optional passwords that have a value at the time the command is executed are set to required and their value unset.

This command does not alter any deployment plans. You must specify passwords in the deployment plan prior to deployment. If you do not provide values in the deployment plan, the deployment plan validation fails.

A.1.8.3 Options

Table A–9 shows the command options for `clearAssemblyPasswords`.

Table A–9 clearAssemblyPasswords options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-name`	n	true	none	Name of a top-level assembly.	Name of a top-level assembly from which to remove passwords.

A.1.8.4 Examples

Here are some command examples.

A.1.8.4.1 clearAssemblyPasswords

```
% abctl clearAssemblyPasswords -name myWlsAssembly
```

A.1.9 connectEndpoints

Details for this command follow.

A.1.9.1 Synopsis

```
$ abctl connectEndpoints -from String -fromOutput String -to String -toInput
String [-force]
```

A.1.9.2 Description

Creates a new connection between an appliance output and an appliance input. The protocols of the output and input must match. The owners of the output and input are not required to be direct siblings. If you specify non-sibling endpoints, the command either locatea existing assembly-level endpoints to use, or automatically creates them as needed to make a valid connection.

A.1.9.3 Options

Table A–10 shows the command options for `connectEndpoints`.

Table A–10 connectEndpoints options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-from`	fr	true	none	<asssembly>/<appliance>	An appliance output to be connected to an appliance input.
`-fromOutput`	fout	true	none	A name.	The name of the appliance-level output to connect from.
`-to`	t	true	none	<asssembly>/<appliance>	An appliance input to be connected to an appliance output.
`-toInput`	tin	true	none	A name.	The name of the appliance-level input to connect to.
`-force`	f	false	none	N/A	If `-force` is set, allows certain connection commands to succeed which would otherwise fail. Specifically it allows an existing connection for a given output to be reconnected to a different input. This flag can force an input change, but not an output change. The command still fails if both input and output are unchanged.

A.1.9.4 Examples

Here are some command examples.

A.1.9.4.1 Create new connection

```
# Creates a new connection
% abctl connectEndpoints -from mySite/ohs1 -fromOutput EMRegistration -to
mySite/wls1/managed1 -toInput Default

Successfully connected from mySite/ohs1:EMRegistration to
mySite/wls1/managed1:Default.
```

A.1.9.4.2 Failure when output is already connected

```
# Command fails because the output is already connected.
% abctl connectEndpoints -from mySite/ohs1 -fromOutput EMRegistration -to
mySite/wls1/AdminServer -toInput Default
```

A.1.9.4.3 Success when using the -force option

```
# Command succeeds because the -force option is used.
./abctl connectEndpoints -from mySite/ohs1 -fromOutput EMRegistration -to
mySite/wls1/AdminServer -toInput Default -force

Successfully connected from mySite/ohs1:EMRegistration to
mySite/wls1/AdminServer:Default.
```

A.1.9.4.4 Failure when output and input are already connected to each other

```
# Command fails because the output and input are already connected to each other.
% abctl connectEndpoints -from mySite/ohs1 -fromOutput EMRegistration -to
mySite/wls1/AdminServer -toInput Default -force
```

A.1.10 copy

Details for this command follow.

A.1.10.1 Synopsis

```
$ abctl copy -name String -copyTo String
```

A.1.10.2 Description

Copy an introspected component - that is, an appliance or an atomic assembly. Appliances inside an atomic assembly may not be copied. External resources may not be copied. The copy is placed at the root of the catalog. Sharing of any captured file sets and/or created virtual machine templates is broken, which means new copies of those things are made as well.

A.1.10.3 Options

Table A–11 shows the command options for `copy`.

Table A–11 copy options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-name`	n	true	none	N/A	Name of an appliance or assembly in the catalog.
`-copyto`	ct	true	none	N/A	The name to use for the new copy of the appliance.

A.1.10.4 Examples

Here are some command examples.

A.1.10.4.1 Copy

```
% abctl copy -name mySite/myWls -copyTo wls2
```

A.1.11 copyAssemblyArchive

Details for this command follow.

A.1.11.1 Synopsis

```
$ abctl copyAssemblyArchive -fileName Path -assemblyName String -connectionName
String [-description String]
```

A.1.11.2 Description

Copies an assembly archive to a Deployer repository. The Deployer copies the assembly archive directly from the specified filesystem location. The specified path must be readable by the Deployer.

A.1.11.3 Options

Table A–12 shows the command options for `copyAssemblyArchive`.

Table A–12 copyAssemblyArchive options

Name	Alias	Req'd	Default Values	Possible Values	Description
-fileName	fn	false	none	A path to the assembly archive file.	The file name of the assembly archive to copy.
-assemblyName	an, n, name	true	none	A string representing the name of the assembly.	The name of the assembly.
-description	d	false	none	A string representing the description of the assembly.	Assembly description.
-connectionName	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.11.4 Examples

Here are some command examples.

A.1.11.4.1 Copy Assembly Archive

```
% abctl copyAssemblyArchive -fileName=c:/mySite.ova -assemblyName TheAssembly
-connectionName myConnection
```

A.1.12 createAssembly

Details for this command follow.

A.1.12.1 Synopsis

```
$ abctl createAssembly -name String [-defaultNetwork String] [-force]
[-description]
```

A.1.12.2 Description

Creates a new assembly in the catalog if one does not already exist by the specified name.

A.1.12.3 Options

Table A–13 shows the command options for `createAssembly`.

Table A–13 createAssembly options

Name	Alias	Req'd	Default Values	Possible Values	Description
-name	n	true	none	Name of the new assembly.	Name of a new assembly to be created.

Table A–13 (Cont.) createAssembly options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-defaultNetwork`	dn	false	none	Name of the default vNet.	If specified, specifies the name to use for the assembly's default vNet.
`-force`	f	false	none	N/A	If `-force` is set, the existing appliance or assembly in the catalog using the same as the newly-created assembly will be overridden.
`-description`	d	false	Name of assembly.	N/A	Descriptive text about the assembly.

A.1.12.4 Examples

Here are some command examples.

A.1.12.4.1 Creating an Assembly

```
$ abctl createAssembly -name myAssembly -defaultNetwork intranet
```

A.1.13 createAssemblyArchive

Details for this command follow.

A.1.13.1 Synopsis

```
$ abctl createAssemblyArchive -name String -platform String [-force] [-quiet]
[-certificateFile Path] [-keyFile Path]
```

A.1.13.2 Description

Creates an assembly archive for the named top-level assembly. This command can only be invoked on a top-level assembly. Additionally, all the sub-appliances within the assembly must previously have been templated using the `createTemplate` command.

Optionally, an assembly archive may be signed, allowing the Deployer to verify its integrity. To sign the archive, specify the `-certificateFile` and `-keyFile` options and respond to the command prompt for the key passphrase.

A.1.13.3 Options

Table A–16 shows the command options for `createAssemblyArchive`.

Table A–14 createAssemblyArchive options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-name`	n	true	none	Name of a top-level (not nested) assembly.	A top-level assembly.
`-platform`	p	true	none	A string representing any valid platform for which templates have been created for the named assembly.	Target platform for which the assembly archive is built.
`-force`	f	false	false	N/A	If specified, any existing assembly archive will be overwritten. If no archive exists, this flag has no effect.

Table A–14 (Cont.) createAssemblyArchive options

Name	Alias	Req'd	Default Values	Possible Values	Description
-quiet	q	false	none	N/A	By default, the command shows detailed progress/success messages. If -quiet is set, the command turns off verbose mode and shows only one or two progress/success messages.
-certificateFile	c	false	none	N/A	PEM encoded X509 certificate file for archive signing.
-keyFile	k	false	none	N/A	PEM encoded RSA private key file for archive signing.

A.1.13.4 Examples

Here are some command examples.

A.1.13.4.1 Creating an Assembly Archive

```
$ abctl createAssemblyArchive -name myWlsAssembly -platform OVM
```

A.1.14 createAssemblyFileSetDefinition

Details for this command follow.

A.1.14.1 Synopsis

```
$ abctl createAssemblyFileSetDefinition -name String -assembly String -mountPoint
Path [-mountType String] [-mountOptions String] [-quotaNumber Numeric] [-quotaUnit
String]
```

A.1.14.2 Description

Creates an Assembly file set definition on the specified assembly. This assembly must be top-level and non-atomic.

A.1.14.3 Options

Table A–15 shows the command options for createAssemblyFileSetDefinition.

Table A–15 createAssemblyFileSetDefinition options

Name	Alias	Req'd	Default Values	Possible Values	Description
-name	n	true	none	N/A	Name of assembly file set definition. By default this will be the name of the mounted volume and the simple name of the .iso file (uploaded within Assembly Resource) that will populate that volume.
-assembly	a	true	none	N/A	Name of top-level, non-atomic assembly.
-mountPoint	mp	true	none	N/A	Directory location where volume will be mounted. Must be an absolute path.
-mountType	mt	false	NFS	Specifies type of mount.	.Specifies type of mount.

Table A–15 (Cont.) createAssemblyFileSetDefinition options

Name	Alias	Req'd	Default Values	Possible Values	Description
-mountOptions	mo	false	oracle	N/A	Specifies mounting options.
-quotaNumber	qn	false	none	N/A	Specifies numeric value of the maximum size this volume can occupy.
-quotaUnit	qu	false	none	Megabytes, Gigabytes, Terabytes	Specifies units associated with the quotaNumber parameter.

A.1.14.4 Examples

Here are some command examples.

A.1.14.4.1 Create Assembly File Set Definition

```
% abctl createAssemblyFileSetDefinition -name foo -assembly mySite -mountPoint
/work/webtier_mt
```

A.1.15 createAssemblyInstance

Details for this command follow.

A.1.15.1 Synopsis

```
$ abctl createAssemblyInstance -deploymentPlan Path -name String -version String
[-target String] -connectionName String
```

A.1.15.2 Description

Creates an assembly instance for an assembly.

A.1.15.3 Options

Table A–16 shows the command options for `createAssemblyInstance`.

Table A–16 createAssemblyInstance options

Name	Alias	Req'd	Default Values	Possible Values	Description
-deploymentPlan	dp	true	none	A file path to the assembly deployment plan on disk.	Specifies a path to a deployment plan file to use for the assembly instance.
-name	n	true	none	A string representing the name of the assembly.	The name of the assembly.
-version	v	true	none	A string representing the version of the assembly.	Assembly version.
-target	t	false	none	A string representing the name of the target.	The name of the target.
-connectionName	c	true	none	A string representing the name of the connection to the Deployer Web Service.	The name of a connection to the Deployer Web Service.

A.1.15.4 Examples

Here are some command examples.

A.1.15.4.1 Creating an Assembly Instance

```
$ abctl createAssemblyInstance -deploymentPlan c:/MyDeploymentPlan.xml -name
MyAssembly -version 1
```

A.1.16 createDeployerConnection

Details for this command follow.

A.1.16.1 Synopsis

```
$ abctl createDeployerConnection -name String -url String [-username String]
[-noReviewCert]
```

A.1.16.2 Description

Creates a new connection between `abctl` and the Deployer. If you specify a connection using the HTTPS protocol, you may be prompted to approve a certificate.

A.1.16.3 Options

Table A–17 shows the command options for `createDeployerConnection`.

Table A–17 createDeployerConnection options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-name`	n	true	none	A string representing the name of the Deployer connection.	The name of the Deployer connection.
`-url`	u	true	none	A string representing the URL of the Deployer Web service.	The URL of the Deployer.
`-username`	un	true	none	A string representing the username.	The username to use to authenticate with the Deployer Web service.
`-noReviewCert`	nr	false	none	N/A	If set, do not present the HTTPS connection certificate for approval.

A.1.16.4 Examples

Here are some command examples.

A.1.16.4.1 Creating a Deployer Connection

```
$ abctl createDeployerConnection -name WLS1 -url http://localhost:7001 -username
cloudAdmin
```

A.1.17 createEMConnection

Details for this command follow.

A.1.17.1 Synopsis

```
$ abctl createEmConnection -connectionURL emMachine:port -connectionUser admin
-namedHostCredential hostCredential -remoteUser mySshUser -remoteWorkingDir
myRemoteWorkingDir [-sshPort port] [-privateKeyFile ~/.ssh/id_rsa]
[remoteEMAgentHost ]
```

A.1.17.2 Description

Creates a connection to an Enterprise Manager Software Library, and persists the connection in a connections file.

You must specify the fully qualified hostname of the remote Enterprise Manager machine, for example `myhost.example.com` instead of `myhost`.

When you perform this command, you are prompted for a connection password.

A.1.17.3 Options

Table A–18 shows the command options for `createEMConnection`.

Table A–18 createEMConnection options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-connectionURL`	c	true	none	Hostname:port.	URL for connecting to EM Software Library.
`-connectionUser`	cu	true	none	Valid EM Software Library User.	User for EM Software Library.
`-namedHostCredential`	n	true	none	Valid Named Host Credential.	Named Host Credential.
`-remoteUser`	ru	true	none	Valid SSH user.	SSH user for connecting to the machine where the EM Software Library is located.
`-sshPort`	s	false	none	Valid SSH port number.	SSH port for EM Software Library machine.
`-privateKeyFile`	pkf	false	none	~/.ssh/id_rsa, id_rsa	Local private SSH key file used for SSH to the remote EM Software Library machine.
`-remoteWorkingDir`	rwd	true	`none`	/scratch,/home/mydir	Valid directory on EM Software Library machine, where assembly archives are uploaded and consumed. Oracle Corporation recommends that you do not use the `/tmp` directory.
`-remoteEmAgentHost`		false	`none`	Hostname or IP address, without http:// or port.	Specify in scenarios where the EM webpage/Web service is on a different machine than where the agent resides.

A.1.17.4 Examples

Here are some command examples.

A.1.17.4.1 createEMConnection

```
$ abctl createEmConnection -connectionURL emMachine:7791 -connectionUser admin
-namedHostCredential hostCredential -remoteUser mySshUser -remoteWorkingDir
/scratch/myovas -sshPort 23 -privateKeyFile ~/.ssh/id_rsa -remoteEmAgentHost
assemblyarchives.reside.here
```

A.1.18 createExternalResources

Details for this command follow.

A.1.18.1 Synopsis

```
$ abctl createExternalResources -from String [-fromOutput String] [-name String]
[-recurse]
```

A.1.18.2 Description

Creates and connects external resources for each of an appliance's or assembly's unconnected outputs. Optionally, specify an individual output for which you want to create an external resource, and a name for the external resource.

Do not specify `-name` to search the top level assembly for an existing external reference that is compatible with the output. When found, a connection is established to the existing external resource instead of creating another external resource with the same properties. You can override this behavior by specifying a name of an external resource that does not yet exist.

Specify `-name` to search the top level assembly for an existing external resource with the specified name. When found, a connection is established to the existing external resource instead of creating another external resource. If you use an existing external resource that does not contain a compatible appliance input, a new appliance input is added to the existing external resource. You can override this behavior by specifying a name of an external resource that does not yet exist.

Recursion, creating external resources for each of an assembly's sub-elements, occurs automatically (the `-recurse` option is no longer necessary) , except when naming a specific appliance output.

Note: Wiring an output to an input on the same appliance is supported.

A.1.18.3 Options

Table A–19 shows the command options for `createExternalResources`.

Table A–19 createExternalResources options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-from`	fr	true	none	Appliance or assembly name.	Name of the appliance or assembly for which you want to external resources.
`-fromOutput`	fout	false	none	Appliance output or assembly output name.	Name of the appliance or assembly output for which you want to create an external resource.
`-name`	n	false	none	New external resource name.	Name of the new external resource appliance. This parameter is only applicable when creating an external resource for an individual output using the `-fromOutput` parameter.
`-recurse`	r	false	none	N/A	If specified, create external resources for each of an assembly's sub-elements.

A.1.18.4 Examples

Here are some command examples.

A.1.18.4.1 Create External Resources for an Assembly

```
% abctl createExternalResources -from mySite/myWls
```

A.1.18.4.2 Create External Resources for Each of an Assembly's Sub-elements

```
% abctl createExternalResources -from mySite -r
```

A.1.18.4.3 Create an External Resource for the jdbc0 Output

```
% abctl createExternalresources -from mySite/myWls -fromOutput jdbc0 -name my_Ext_
JDBC
```

A.1.19 createInterface

Details for this command follow.

A.1.19.1 Synopsis

```
$ abctl createInterface -name String -appliance String [-default]
```

A.1.19.2 Description

Creates a new physical network interface on the specified appliance.

A.1.19.3 Options

Table A–20 shows the command options for `createInterface`.

Table A–20 createInterface options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-name`	n	true	none	N/A	The name of the new interface to create.
`-appliance`	a	true	none	N/A	The name of an existing appliance.
`-default`	d	true	none	N/A	If set, the new interface will be marked as the default interface for the appliance.

A.1.19.4 Examples

Here are some command examples.

A.1.19.4.1 Create Interface

```
% abctl createInterface -name myInterface -appliance mySite/myOhs -default
```

A.1.20 createNativeFileSetDefinition

Details for this command follow.

A.1.20.1 Synopsis

```
$ abctl createNativeFileSetDefinition -name String -appliance String [-shared]
-rootDirectory Path [-excludes String...] [-noCapture] [-osOwner String] [-osGroup
String] [-freeSpaceNumber Numeric] [-freeSpaceUnit String]
```

A.1.20.2 Description

Creates a Native file set definition and adds it to a top-level or nested appliance. Use this command to define a "local Linux" or "shared Linux" file set.

A.1.20.3 Options

Table A–21 shows the command options for `createNativeFileSetDefinition`.

Table A–21 createNativeFileSetDefinition options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-name`	n	true	none	N/A	Name of the file set definition.
`-appliance`	a	true	none	N/A	Name of top-level or nested appliance in which definition is being added.
`-rootDirectory`	rd	true	none	N/A	Top-level directory for all files included in the file set. Path must be absolute.
`-excludes`	e	false	none	N/A	One or more files and/or directories to be excluded from file set. Paths must be relative to the `rootDirectory` location.

Table A–21 (Cont.) createNativeFileSetDefinition options

Name	Alias	Req'd	Default Values	Possible Values	Description
-shared	s	false	false	N/A	Specifies whether file set will be stored on a 'shared' file system. Omitting this parameter allows the file set to be stored on the local file system.
-noCapture	nc	false	false	N/A	Indicates whether source files of this definition should be included when capturing file sets. Without this argument, the definition's file set will be captured.
-osOwner	oo	false	oracle	N/A	Owner name to be assigned files in assembly instance.
-osGroup	og	false	oracle	N/A	Group name to be assigned files in assembly instance.
-freeSpaceNumber	fsn	false	Percent	Gigabytes, Megabytes, Percent	Specifies numeric value of extra space to be allocated when the file set is staged. *Note*: The default ('50 Percent') applies when neither free space parameter is specified. When one of these Free Space parameters is provided in command then the other is required.
-freeSpaceUnit	fsu	false	50	N/A	Specifies units associated with the freeSpaceNumber parameter. It must be specified if and only if the freeSpaceNumber parameter is used.

A.1.20.4 Examples

Here are some command examples.

A.1.20.4.1 Create Native File Set Definition

```
% abctl createNativeFileSetDefinition -name OhsOracleHome -appliance mySite/ohs1
-shared -rootDirectory /scratch/aime1/middleware -excludes wlserver_10.3 jrockit*
jdk* user_projects
```

A.1.21 createNFSFileSetDefinition

Details for this command follow.

A.1.21.1 Synopsis

```
$ abctl createNFSFileSetDefinition -name String -appliance String -rootDirectory
Path [-excludes String...] [-noCapture] [-osOwner String] [-osGroup String]
```

A.1.21.2 Description

Creates an NFS file set definition and adds it to a top-level or nested appliance.

A.1.21.3 Options

Table A–22 shows the command options for `createNativeFileSetDefinition`.

Table A–22 createNFSFileSetDefinition options

Name	Alias	Req'd	Default Values	Possible Values	Description
-name	n	true	none	N/A	Name of the file set definition.
-appliance	a	true	none	N/A	Name of top-level or nested appliance in which definition is being added.
-rootDirectory	rd	true	none	N/A	Top-level directory for all files included in the file set. Path must be absolute.

Table A–22 (Cont.) createNFSFileSetDefinition options

Name	Alias	Req'd	Default Values	Possible Values	Description
-excludes	e	false	none	N/A	One or more files and/or directories to be excluded from file set. Paths must be relative to the rootDirectory location.
-noCapture	nc	false	false	N/A	Indicates whether source files of this definition should be included when capturing file sets. Without this argument, the definition's file set will be captured.
-osOwner	oo	false	oracle	N/A	Owner name to be assigned files in assembly instance.
-osGroup	og	false	oracle	N/A	Group name to be assigned files in assembly instance.

A.1.21.4 Examples

Here are some command examples.

A.1.21.4.1 Create NFS File Set Definition

```
% abctl createNFSFileSetDefinition -name OhsOracleHome -appliance mySite/ohs1 \
-rootDirectory /work/perl-scripts
```

A.1.22 createRAWFileSetDefinition

Details for this command follow.

A.1.22.1 Synopsis

```
$ abctl createRAWFileSetDefinition -name String -appliance String -rootDirectory
Path [-osOwner String] [-osGroup String] [-freeSpaceNumber Numeric]
[-freeSpaceUnit String]
```

A.1.22.2 Description

Creates a RAW file set definition and adds it to a top-level or nested appliance.

A.1.22.3 Options

Table A–23 shows the command options for createRAWFileSetDefinition.

Table A–23 createRAWFileSetDefinition options

Name	Alias	Req'd	Default Values	Possible Values	Description
-name	n	true	none	N/A	Name of the file set definition.
-appliance	a	true	none	N/A	Name of top-level or nested appliance in which definition is being added.
-rootDirectory	rd	true	none	N/A	Top-level directory for all files included in the file set. Path must be absolute.
-osOwner	oo	false	oracle	N/A	Owner name to be assigned files in assembly instance.

Table A–23 (Cont.) createRAWFileSetDefinition options

Name	Alias	Req'd	Default Values	Possible Values	Description
-osGroup	og	false	oracle	N/A	Group name to be assigned files in assembly instance.
-freeSpaceNumber	fsn	false	Percent	Gigabytes, Megabytes, Percent	Specifies numeric value of extra space to be allocated when the file set is staged. *Note*: The default ('50 Percent') applies when neither free space parameter is specified. When one of these Free Space parameters is provided in command then the other is required.
-freeSpaceUnit	fsu	false	50	N/A	Specifies units associated with the freeSpaceNumber parameter. It must be specified if and only if the freeSpaceNumber parameter is used.

A.1.22.4 Examples

Here are some command examples.

A.1.22.4.1 Create RAW File Set Definition

```
% abctl createRAWFileSetDefinition -name DbHome -appliance mySite/db1
-rootDirectory /work/dbms
```

A.1.23 createTags

Details for this command follow.

A.1.23.1 Synopsis

```
$ abctl createTags -tag String... -resource String... -connectionName String
```

A.1.23.2 Description

Creates one or more tags for a resource.

A.1.23.3 Options

Table A–24 shows the command options for `createTags`.

Table A–24 createTags options

Name	Alias	Req'd	Default Values	Possible Values	Description
-tag	t	true	none	A set of name=value pairs specifying the tags.	Specifies one or more tags to tag a resource with.
-resource	r	true	none	A string specifying the resource id of the object to tag.	Specifies one or more resources to apply a tag to.
-connectionName	c	true	none	A string representing the name of the connection to the Deployer Web Service.	The name of a connection to the Deployer Web Service.

A.1.23.4 Examples

Here are some command examples.

A.1.23.4.1 Create Tags

```
$ abctl createTags -tag key=value -resource MyResource
```

A.1.24 createTarget

Details for this command follow.

A.1.24.1 Synopsis

```
$ abctl createTarget -name String -type String -connectionName String [-properties
String...] [-default]
```

A.1.24.2 Description

Creates a deployment target. This command is enabled for Oracle VM targets.

The following are required and optional properties for the `createTarget` command. (The asterisks (*) indicate a required property):

- ovm [ovm.url*, ovm.poolName*, ovm.user*, ovm.pwd*, ovm.vmmversion*, ovm.vmOperationTimeout]

A.1.24.3 Oracle VM Configuration

Oracle recommends that you configure your target connections for Oracle VM 3 with TCP instead of HTTP protocol.

To configure with TCP, specify a URL of the form "tcp://their-ovm-host:54321".

A.1.24.4 Options

Table A–25 shows the command options for `createTarget`.

Table A–25 createTarget options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-name`	n	true	none	A string representing the name of the target.	The name of the target.
`-type`	t	true	none	A string representing the type of target. Possible value is `ovm`.	The type of assembly instance target.
`-properties`	p	false	none	A string representing property=value pairs to set on the target.	The properties to set on the target.
`-default`	d	false	false	N/A	If set, indicates that this target is the default target.
`-connection Name`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.24.5 Examples

Here are some command examples.

A.1.24.5.1 Create Target

```
$ abctl createTarget -name MyTarget -type OVM -properties ovm.poolName=my_pool
ovm.vmOperationTimeout=1200 ovm.vmmversion=3.0 ovm.user=admin ovm.pwd
ovm.url=tcps://my.host.com:54322 -connectionName myDeployerConn
```

A.1.25 createTemplate

Details for this command follow.

A.1.25.1 Synopsis

```
$ abctl createTemplate -name String -platform String [-quiet] [-baseImage Path]
[-force]
```

A.1.25.2 Description

Creates an appliance template for a given appliance or assembly.

A.1.25.3 Options

Table A–26 shows the command options for `createTemplate`.

Table A–26 createTemplate options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-baseImage`	bi	false	The following is the order of precedence for base image detection: 1. The location specified by the `-baseImage` flag 2. `$AB_INSTANCE/templates/baseImage/OVM/OEL` (for OVM) 3. `$ORACLE_HOME/templates/baseImage/OVM/OEL` (for OVM)	Path to a valid base image.	Path to a valid base image used to create an appliance template.
`-force`	f	false	false	N/A	If `-force` is set, the existing template(s) for assemblies or appliances will be overridden. If the template does not exist, this flag has no effect.
`-name`	n	true	none	Name of appliance or assembly in catalog. Nested appliances or assemblies are referred to with a slash ("/"). For example: mySite/myOhs.	Name of an appliance or assembly in the catalog.
`-platform`	p	true	none	OVM	Target platform for which the appliance template is built.
`-quiet`	q	false	false	N/A	By default, the command shows detailed progress/success messages. If `-quiet` is set, the command turns off verbose mode and shows only one or two progress/success messages.
`-validateBaseImage`	v	false	none	N/A	If `-validateBaseImage` is set, changes operation from creating a template to validating the base image. Does not prompt for a template password. The validation output is more verbose.

A.1.25.4 Examples

Examples for this command follow.

A.1.25.4.1 No valid base image is found

```
$ abctl createTemplate -name myOhs -platform OVM
Executing createTemplate command.
Error: OAB-7389: Failed to create VM template for myOhs.
```

```
Caused by: OAB-20343: Unable to locate a valid default base image.
  Action: Specify a base image location, or place a base image in default
location. Refer to user guide for detail.
```

A.1.25.4.2 Template already exists for given OS type

```
$ abctl createTemplate -name myOhs -platform OVM
Executing createTemplate command.
Error: OAB-7389: Failed to create VM template for myOhs.
Caused by: OAB-20120: Appliance myOhs already has template for OEL.
  Action: Use -force flag to override existing template.
```

A.1.25.4.3 Successful Template Creation

```
$ abctl createTemplate -name myOhs -platform OVM -baseImage
/private/baseImage/OVM/OEL/System.img
Executing createTemplate command.
Set the root and vnc passwords that will be configured in the template.
Enter root password:
Retype root password:
Enter vnc password:
Retype vnc password:
  Step 1 of 2: Creating template for appliance myOhs started.
    Step 1 of 4: Copying base image to catalog started.
    Step 2 of 4: Copying base image to catalog completed.
    Step 3 of 4: Creating product disk for myOhs_root started.
    Step 4 of 4: Creating product disk for myOhs_root completed.
  Step 2 of 2: Creating template for appliance myOhs completed.
Successfully created template for myOhs.
```

A.1.26 createVnet

Details for this command follow.

A.1.26.1 Synopsis

```
$ abctl createVnet -name String -assembly String
```

A.1.26.2 Description

Creates a new Vnet within the specified assembly.

A.1.26.3 Options

Table A–27 shows the command options for `createVnet`.

Table A–27 createVnet options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-name`	n	true	none	N/A	The name of the new Vnet to create.
`-assembly`	a	true	none	N/A	The name of an existing top-level assembly.

A.1.26.4 Examples

Here are some command examples.

A.1.26.4.1 Create Vnet

```
% abctl createVnet -name myVnet -assembly mySite
```

A.1.27 delete

Details for this command follow.

A.1.27.1 Synopsis

```
$ abctl delete [-name] String [-archiveOnly]
```

A.1.27.2 Description

Deletes the appliance or assembly with the given name. Only the top-level appliance or assembly can be deleted. Nested appliances or assemblies cannot be deleted using this command. Also, top level appliances or assemblies that share captured file sets and/or templates for an assembly that has an archive cannot be deleted.

A.1.27.3 Options

Table A–28 shows the command options for `delete`.

Table A–28 delete options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-name`	n	true	none	Name of the top-level appliance or assembly.	Name of the top-level appliance or assembly to be deleted.
`-archiveOnly`	o	false	none	N/A	If specified, delete only the assembly archive, leaving the rest of the assembly intact.

A.1.27.4 Examples

Here are some command examples.

A.1.27.4.1 Attempted to delete nested appliance/assembly

```
$ abctl delete -name mySite/myOhs
Executing delete command.
Error: OAB-7672: Unable to delete mySite/myOhs from catalog.
  Cause: Nested appliance or assembly cannot be deleted.
  Action: Use AbStudio (GUI) to delete nested appliances or assemblies.
```

A.1.27.4.2 Successful Delete

```
$ abctl delete -name myOhs
Executing delete command.
Successfully deleted myOhs.
```

A.1.27.4.3 Delete of Only an Archive

```
$ abctl delete -name myOhs -archiveOnly
```

A.1.27.4.4 Delete Failed

```
$ abctl delete -name myOhs
Deleted metadata
Deleted File Sets
Error: Failed to delete templates.
```

A.1.28 deleteAssemblyArchive

Details for this command follow.

A.1.28.1 Synopsis

```
$ abctl deleteAssemblyArchive -name String [-version String] -connectionName
String
```

A.1.28.2 Description

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

A.1.28.3 Options

Table A–29 shows the command options for `deleteAssemblyArchive`.

Table A–29 deleteAssemblyArchive options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-name`	n	true	none	A string representing the name of the assembly.	Name of an assembly in the Deployer repository.
`-version`	v	false	The default is the latest version number assigned by the Deployer.	A string representing the version of the assembly.	Specifies the version of the assembly to delete from the Deployer repository.
`-connection Name`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.28.4 Examples

Here are some command examples.

A.1.28.4.1 Deleting an Assembly Archive

```
$ abctl deleteAssemblyArchive -name TheAssemblyArchive -version 1
```

A.1.29 deleteAssemblyInstance

Details for this command follow.

A.1.29.1 Synopsis

```
$ abctl deleteAssemblyInstance -assemblyInstanceId String -connectionName String
```

A.1.29.2 Description

Deletes an assembly instance.

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

A.1.29.3 Options

Table A–30 shows the command options for `deleteAssemblyInstance`.

Table A–30 deleteAssemblyInstance options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-assemblyInstanceId`	d	true	none	A string representing the `assemblyInstanceId`.	The identifier of an assembly instance to be deleted.
`-connectionName`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.29.4 Examples

Here are some command examples.

A.1.29.4.1 Deleting an Assembly Instance

```
$ abctl deleteAssemblyInstance -assemblyInstanceId MyId -c myDeployerConn
```

A.1.30 deleteAssemblyResources

Details for this command follow.

A.1.30.1 Synopsis

```
$ abctl deleteAssemblyResources -assemblyName String -connectionName String
[-version String] [-fileName String...]
```

A.1.30.2 Description

Deletes an assembly resources file for an existing assembly.

A.1.30.3 Options

Table A–31 shows the command options for `deleteAssemblyResources`.

Table A–31 deleteAssemblyResources options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-assemblyName`	n	true	none	A string representing the name of the assembly.	The name of the assembly.
`-connectionName`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
`-version`	v	false	none	A string representing the assembly version.	The assembly version.
`-fileName`	fn	true	none	A string representing the file path to the assembly resources files on disk.	Uploads an assembly resources file to the Deployer.

A.1.30.4 Examples

Here are some command examples.

A.1.30.4.1 Deleting an Assembly Resources File

```
$ abctl deleteAssemblyResources -assemblyName FOO -version 1
```

A.1.31 deleteDeployerConnection

Details for this command follow.

A.1.31.1 Synopsis

```
$ abctl deleteDeployerConnection [-name] String
```

A.1.31.2 Description

Deletes a connection to the Deployer, and removes the connection from the connections file.

A.1.31.3 Options

Table A–32 shows the command options for `deleteDeployerConnection`.

Table A–32 deleteDeployerConnection options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-name`	n	true	none	A string representing the name of the Deployer connection.	The name of the Deployer connection.

A.1.31.4 Examples

Here are some command examples.

A.1.31.4.1 Deleting a Connection to the Deployer

```
$ abctl deleteDeployerConnection -name WLS1
```

A.1.32 deleteDeploymentPlan

Details for this command follow.

A.1.32.1 Synopsis

```
$ abctl deleteDeploymentPlan -assemblyName String -planName String -connectionName
String [-version String]
```

A.1.32.2 Description

Deletes a deployment plan from an existing assembly.

A.1.32.3 Options

Table A–33 shows the command options for `deleteDeploymentPlan`.

Table A–33 deleteDeploymentPlan options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-assemblyName`	n	true	none	A string representing the name of the assembly.	The name of the assembly.
`-planName`	p	true	none	A string representing the name of the deployment plan.	The name of the deployment plan.
`-connectionName`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
`-version`	v	false	none	A string representing the assembly version.	The assembly version.

A.1.32.4 Examples

Here are some command examples.

A.1.32.4.1 Delete Deployment Plan

```
$ abctl deleteDeploymentPlan -assemblyName FOO -version 1 -planName FOO
```

A.1.33 deleteEMAssemblyArchive

Details for this command follow.

A.1.33.1 Synopsis

```
$ abctl deleteEMAssemblyArchive -name String -version String
```

A.1.33.2 Description

Deletes the specified version of an assembly archive from the Enterprise Manager Software Library.

A.1.33.3 Options

Table A–34 shows the command options for `deleteEMAssemblyArchive`.

Table A–34 deleteEMAssemblyArchive options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-name`	n	true	none	Name of assembly archive	Name of assembly archive to delete from the EM Software Library.
`-version`	v	true	none	1.0, 1.2, 2.0, etc.	Version of an assembly archive to delete from the EM Software Library.

A.1.33.4 Examples

Here is a command example.

A.1.33.4.1 Delete EM Assembly Archive

```
% abctl deleteEMAssemblyArchive -name archiveName -version 1.2
```

A.1.34 deleteEMConnection

Details for this command follow.

A.1.34.1 Synopsis

```
$ abctl deleteEMConnection
```

A.1.34.2 Description

Deletes the connection to the EM Software Library.

A.1.34.3 Options

None.

A.1.34.4 Examples

Here are some command examples.

A.1.34.4.1 Deleting an EM Connection

```
$ abctl deleteEMConnection
```

A.1.35 deleteFailedApplianceInstances

Details for this command follow.

A.1.35.1 Synopsis

```
$ abctl deleteFailedApplianceInstances -applianceId String -applianceInstanceIds
String... -connectionName String
```

A.1.35.2 Description

Deletes one or more appliance instances in the failed state.

A.1.35.3 Options

Table A–35 shows the command options for `deleteFailedApplianceInstances`.

Table A–35 deleteFailedApplianceInstances options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-applianceId`	a	true	none	A string representing the ID of the appliance.	The ID of the appliance.
`-applianceInstanceIds`	ai	true	none	A string representing the IDs of the appliances.	The IDs of the appliance instances in a failed state to delete.
`-connectionName`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.35.4 Examples

Here is a command example.

A.1.35.4.1 Delete Failed Appliance Instances

```
% abctl deleteFailedApplianceInstances -applianceId MyId -applianceInstanceIds
MyInstanceId1
```

A.1.36 deleteFileSetDefinition

Details for this command follow.

A.1.36.1 Synopsis

```
$ abctl deleteFileSetDefinition -name String -appliance String
```

A.1.36.2 Description

Deletes a file set definition from a top-level or nested appliance.

A.1.36.3 Options

Table A–36 shows the command options for `deleteFileSetDefinition`.

Table A–36 deleteFileSetDefinition options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-name`	n	true	none	N/A	Name of the file set definition.
`-appliance`	a	true	none	N/A	Name of top-level or nested appliance in which definition is being added.

A.1.36.4 Examples

Here are some command examples.

A.1.36.4.1 Delete File Set Definition

```
% abctl deleteFileSetDefinition -name OhsOracleHome -appliance mySite/ohs1
```

A.1.37 deleteInterface

Details for this command follow.

A.1.37.1 Synopsis

```
$ abctl deleteInterface -name String -appliance String
```

A.1.37.2 Description

Deletes an existing physical network interface from the specified appliance.

A.1.37.3 Options

Table A–37 shows the command options for `deleteInterface`.

Table A–37 deleteInterface options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-name`	n	true	none	N/A	The name of the interface to delete.
`-appliance`	a	true	none	N/A	The name of an existing appliance.

A.1.37.4 Examples

Here are some command examples.

A.1.37.4.1 Delete Interface

```
% abctl deleteInterface -name myInterface -appliance mySite/myOhs
```

A.1.38 deleteLogEvents

Details for this command follow.

A.1.38.1 Synopsis

```
$ abctl deleteLogEvents -connectionName String [-user String] [-assemblyInstanceId
String] [-afterTime String] [-beforeTime String]
```

A.1.38.2 Description

Deletes log events based on specified input criteria.

A.1.38.3 Options

Table A–38 shows the command options for `deleteLogEvents`.

Table A–38 deleteLogEvents options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-connectionName`	c	true	none	A string representing the user name.	Specifies the user name to query upon.
`-user`	u	false	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
`-assemblyInstanceId`	a	false	none	A string representing the `assemblyInstanceId`.	Specifies the identifier of the assembly instance to query upon.
`-afterTime`	at	false	none	A string representing the start time.	Specifies the start time of the log event query time frame.
`-beforeTime`	bt	false	none	A string representing the end time.	Specifies the end time of the log event query time frame.

A.1.38.4 Examples

Here are some command examples.

A.1.38.4.1 Delete Log Events

```
$ abctl deleteLogEvents -user ovab -assemblyInstanceId MyId -afterTime 2012-03-01
-beforeTime 2012-03-19
```

A.1.39 deleteRequests

Details for this command follow.

A.1.39.1 Synopsis

```
$ abctl deleteRequests [-requestId String...] -connectionName String
```

A.1.39.2 Description

Deletes one or more previously completed requests.

A.1.39.3 Options

Table A–100 shows the command options for `deleteRequests`.

Table A–39 deleteRequests options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-connectionName`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
`-requestId`	d	false	none	A string representing the `requestId`.	The `requestId` of a previously completed request.

A.1.39.4 Examples

Here are some command examples.

A.1.39.4.1 Delete Requests

```
$ abctl deleteRequests
```

A.1.40 deleteTags

Details for this command follow.

A.1.40.1 Synopsis

```
$ abctl deleteTags -tag String... -resource String... -connectionName String
```

A.1.40.2 Description

This command deletes one or more tags for a resource.

A.1.40.3 Options

Table A–40 shows the command options for `deleteTags`.

Table A–40 deleteTags options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-connection Name`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
`-resource`	r	true	none	A string specifying the resource id of the object to remove the tag from.	Specifies one or more resources to remove a tag from.
`-tag`	t	true	none	A set of name=value pairs specifying the tags.	Specifies one or more tags to remove from a resource.

A.1.40.4 Examples

Here are some command examples.

A.1.40.4.1 Deleting Tags

```
$ abctl deleteTags -tag foo -resource MyResource
```

A.1.41 deleteTarget

Details for this command follow.

A.1.41.1 Synopsis

```
$ abctl deleteTarget -name String -connectionName String
```

A.1.41.2 Description

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

A.1.41.3 Options

Table A–41 shows the command options for `deleteTarget`.

Table A–41 deleteTarget options

Name	Alias	Req'd	Default Values	Possible Values	Description
-name	n	true	none	A string representing the name of the target.	The name of the target.
-connection Name	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.41.4 Examples

Here are some command examples.

A.1.41.4.1 Deleting a Target

```
$ abctl deleteTarget -name MyTarget
```

A.1.42 deleteVnet

Details for this command follow.

A.1.42.1 Synopsis

```
$ abctl
```

A.1.42.2 Description

Deletes an existing Vnet from the specified assembly.

A.1.42.3 Options

Table A–42 shows the command options for `deleteVnet`.

Table A–42 deleteVnet options

Name	Alias	Req'd	Default Values	Possible Values	Description
-name	n	true	none	N/A	The name of the existing Vnet to delete.
-assembly	a	true	none	N/A	The name of an existing top-level assembly.

A.1.42.4 Examples

Here are some command examples.

A.1.42.4.1 Delete Vnet

```
% abctl deleteVnet -name myVnet -assembly mySite
```

A.1.43 deployAssemblyInstance

Details for this command follow.

A.1.43.1 Synopsis

```
$ abctl deployAssemblyInstance -assemblyInstanceId String -connectionName String
[-waitForComplete] [-pollTime String]
```

A.1.43.2 Description

This command deploys an assembly archive.

A.1.43.3 Options

Table A–43 shows the command options for `deployAssemblyInstance`.

Table A–43 deployAssemblyInstance options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-assemblyInstanceId`	a	true	none	A string representing the `assemblyInstanceId`.	The identifier of an assembly instance to be deployed.
`-connectionName`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
`-waitForComplete`	w	false	no	N/A	Specifies whether to wait for an asynchronous operation to complete successfully.
`-pollTime`	pt	false	5	A string representing the number of seconds.	Specifies the amount of time to wait for an asynchronous operation to complete successfully.

A.1.43.4 Examples

Here are some command examples.

A.1.43.4.1 Deploying an Assembly Instance

```
$ abctl deployAssemblyInstance -assemblyInstanceId MyId
```

A.1.44 describeApplianceInstanceMetrics

Details for this command follow.

A.1.44.1 Synopsis

```
$ abctl describeApplianceInstanceMetrics -applianceInstanceId String
-connectionName String [-average] [-begin String] [-end String] [-metric
String...]

$ abctl describeApplianceInstanceMetrics -assemblyInstanceId String -appliancePath
String -connectionName String [-average] [-begin String] [-end String] [-metric
String...]
```

A.1.44.2 Description

Describes vserver metrics based on specified input criteria.

A.1.44.3 Options

Table A–44 shows the command options for `describeApplianceInstanceMetrics`.

Table A–44 describeApplianceInstanceMetrics options

Name	Alias	Req'd	Default Values	Possible Values	Description
-applianceInstanceId	aii	true *	none	A string representing the applianceId.	One or more applianceIds. * Only one of the following may be specified: applianceInstanceId, assemblyInstanceId.
-assemblyInstanceId	as	true *	none	A string representing the assemblyInstanceId.	One or more assemblyInstanceIds. * Only one of the following may be specified: applianceInstanceId, assemblyInstanceId.
-appliancePath	ap	true *	none	A string representing the appliancePath.	The appliance path of an assembly instance to query upon. * Must be specified when specifying one of the following: assemblyInstanceId
-average	a	false	none	N/A	If specified, the metrics will be averaged.
-begin	b	false	none	A string representing the start time.	Specify the start time to query the metrics.
-end	e	false	none	A string representing the end time.	Specify the end time to query the metrics.
-metric	m	false	none	A set of vserver metrics.	Specify one or more metric to query.
-connectionName	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.44.4 Examples

Here are some command examples.

A.1.44.4.1 Describe Appliance Instance Metrics for a Specified assemblyInstanceId

```
$ abctl describeApplianceInstanceMetrics -assemblyInstanceId myId -appliancePath
myPath -connectionName myConnection -metric cpu memory
```

A.1.44.4.2 Describe Appliance Instance Metrics for a Specified applianceInstanceId

```
$ abctl describeApplianceInstanceMetrics -applianceInstanceId myId -connectionName
myConnection -metric cpu memory
```

A.1.45 describeApplianceInstances

Details for this command follow.

A.1.45.1 Synopsis

```
$ abctl describeApplianceInstances [-assemblyInstanceId String...]
[-applianceInstanceId String...] [-applianceIndex String...] -connectionName
String
```

A.1.45.2 Description

Describes one or more deployed instances of an assembly.

A.1.45.3 Options

Table A–45 shows the command options for describeApplianceInstances.

Table A–45 describeApplianceInstances options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-assemblyInstanceId`	as	false	none	A string representing the assemblyInstanceId.	One or more `assemblyInstanceIds`.
`-applianceInstanceId`	ap	false	none	A string representing the `applianceId`.	One or more `applianceIds`.
`-applianceIndex`	ai	false	none	A string representing the `instanceId`.	One or more `instanceIds`.
`-connectionName`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.45.4 Examples

Here are some command examples.

A.1.45.4.1 Describe Appliance Instances

```
$ abctl describeApplianceInstances
```

A.1.46 describeApplianceMetrics

Details for this command follow.

A.1.46.1 Synopsis

```
$ abctl describeApplianceMetrics
```

A.1.46.2 Description

Describes metrics for the appliance.

The "memory" value displayed is not the *free* memory of an appliance instance's VM, but rather the total memory on the VM, which does not change during the lifespan of the VM.

A.1.46.3 Options

Table A–46 shows the command options for `describeApplianceMetrics`.

Table A–46 describeApplianceMetrics options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-assemblyInstanceId`	as	false	none	A string representing the assemblyInstanceId.	One or more `assemblyInstanceIds`.

Table A–46 (Cont.) describeApplianceMetrics options

Name	Alias	Req'd	Default Values	Possible Values	Description
-applianceInstanceId	ap	false	none	A string representing the applianceId.	One or more applianceIds.
-applianceIndex	ai	false	none	A string representing the instanceId.	One or more instanceIds.
-connectionName	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.46.4 Examples

Here are some command examples.

A.1.46.4.1 Describe Appliance Metrics

```
$ abctl describeApplianceMetrics
```

A.1.47 describeAssemblyArchives

Details for this command follow.

A.1.47.1 Synopsis

```
$ abctl describeAssemblyArchives [-assembly String...] -connectionName String
```

A.1.47.2 Description

Describes one or more assembly archives in the Deployer.

A.1.47.3 Options

Table A–47 shows the command options for describeAssemblyArchives.

Table A–47 describeAssemblyArchives options

Name	Alias	Req'd	Default Values	Possible Values	Description
-assembly	a	false	none	A list of the assemblies to describe.	Specifies the assemblies to describe.
-connectionName	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.47.4 Examples

Here are some command examples.

A.1.47.4.1 Describe Assembly Archives

```
$ abctl describeAssemblyArchives -assembly TheAssembly
```

A.1.48 describeAssemblyInstances

Details for this command follow.

A.1.48.1 Synopsis

```
$ abctl describeAssemblyInstances [-assemblyInstanceId String...] -connectionName
String
```

A.1.48.2 Description

Describes one or more assembly instances.

A.1.48.3 Options

Table A–48 shows the command options for `describeAssemblyInstances`.

Table A–48 describeAssemblyInstances options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-assemblyInstanceId`	a	false	none	A comma-separated list of assembly instance IDs.	Identifiers of one or more assembly instances to be described.
`-connectionName`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.48.4 Examples

Here are some command examples.

A.1.48.4.1 Describe Assembly Instances

```
$ abctl describeAssemblyInstances
```

A.1.49 describeAssemblyResources

Details for this command follow.

A.1.49.1 Synopsis

```
$ abctl describeAssemblyResources -assemblyName String -connectionName String
[-version String]
```

A.1.49.2 Description

Describe assembly resources for an existing assembly

A.1.49.3 Options

Table A–49 shows the command options for `describeAssemblyResources`.

Table A–49 describeAssemblyResources options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-assemblyName`	n	true	none	A string representing the name of the assembly.	The name of the assembly.
`-connectionName`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
`-version`	v	false	none	A string representing the assembly version.	The assembly version.

A.1.49.4 Examples

Here are some command examples.

A.1.49.4.1 Describe Assembly Resources

```
$ abctl describeAssemblyResources -assemblyName FOO -version 1
```

A.1.50 describeAssemblyUsers

Details for this command follow.

A.1.50.1 Synopsis

```
$ abctl describeAssemblyUsers -assembly String -connectionName String
```

A.1.50.2 Description

This command describes one or more users of an assembly.

A.1.50.3 Options

Table A–48 shows the command options for `describeAssemblyUsers`.

Table A–50 describeAssemblyUsers options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-assembly`	a	false	none	A string representing the name of the assembly.	Specifies the assembly whose users will be described.
`-connection Name`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.50.4 Examples

Here are some command examples.

A.1.50.4.1 Describe Assembly Users

```
$ abctl describeAssemblyUsers -assembly MyAssembly
```

A.1.51 describeAssemblyVnets

Details for this command follow.

A.1.51.1 Synopsis

```
$ abctl describeAssemblyVnets -name String
```

A.1.51.2 Description

Describes the logical Vnets defined in a top-level assembly.

A.1.51.3 Options

Table A–51 shows the command options for `describeAssemblyVnets`.

Table A–51 describeAssemblyInstances options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-name`	n	true	none	Name of an assembly.	Name of a top-level, non-atomic assembly whose Vnets are listed.

A.1.51.4 Examples

Here are some command examples.

A.1.51.4.1 Describe Assembly Instances

```
$ abctl describeAssemblyVnets -name mySite
```

A.1.52 describeCatalog

Details for this command follow.

A.1.52.1 Synopsis

```
$ abctl describeCatalog [-name] String] [-long]
```

A.1.52.2 Description

Lists appliances and assemblies in the catalog, and display file set information.

A.1.52.3 Options

Table A–52 shows the command options for `describeCatalog`.

Table A–52 describeCatalog options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-long`	l	false	N/A	N/A	Lists information with maximum detail. Included for compatibility only.
`-name`	n	false	None.	Name of an appliance or assembly. Nested appliances or assemblies are referred to with slash ('/'), for example: `mySite/myOhs`.	If not specified, all appliances and assemblies in the catalog are displayed. If the name of an assembly is specified, its sub-appliances and sub-assemblies are listed in addition to the assembly itself. If the name of an appliance is specified, only that appliance is listed.

A.1.52.4 Examples

Here are some command examples.

A.1.52.4.1 Describe Catalog in Long Format

```
$ abctl describeCatalog -name myAssembly -long
```

A.1.53 describeDeployer

Details for this command follow.

A.1.53.1 Synopsis

```
$ abctl describeDeployer -connectionName String
```

A.1.53.2 Description

Describes a Deployer instance.

A.1.53.3 Options

Table A–53 shows the command options for `describeDeployer`.

Table A–53 describeDeployer options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-connection Name`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.53.4 Examples

Here are some command examples.

A.1.53.4.1 Describe Deployer

```
$ abctl describeDeployer
```

A.1.54 describeDeployerConnections

Details for this command follow.

A.1.54.1 Synopsis

```
$ abctl describeDeployerConnections
```

A.1.54.2 Description

Describes the configured Deployer connections.

A.1.54.3 Options

None.

A.1.54.4 Examples

Here are some command examples.

A.1.54.4.1 Describe Deployer Connections

```
$ abctl describeDeployerConnections
```

A.1.55 describeDeploymentPlans

Details for this command follow.

A.1.55.1 Synopsis

```
$ abctl describeDeploymentPlans [-name String] [-plan String] [-long]
```

A.1.55.2 Description

Describes the available deployment plans.

A.1.55.3 Options

Table A–53 shows the command options for `describeDeploymentPlans`.

Table A–54 describeDeploymentPlan options

Name	Alias	Req'd	Default Values	Possible Values	Description
-name	n	false	.*	Any regular expression.	Name of an assembly or assemblies, specified as a regular expression.
-plan	p	false	.*	Any regular expression.	Name of a plan or plans. It is specified as a regular expression.
-long	l	false	none	N/A	Flag to indicate if the long version of information is required.

A.1.55.4 Examples

Here are some command examples.

A.1.55.4.1 Describe Deployment Plans

```
$ abctl describeDeploymentPlans -name myAssembly -plan myPlan -long
```

A.1.56 describeEMAssemblyArchives

Details for this command follow.

A.1.56.1 Synopsis

```
$ abctl describeEMAssemblyArchives [-name nameOfAssemblyArchive]
```

A.1.56.2 Description

Describes assembly archives in the EM Software Library.

A.1.56.3 Options

Table A–53 shows the command options for `describeEMAssemblyArchives`.

Table A–55 describeEMAssemblyArchives options

Name	Alias	Req'd	Default Values	Possible Values	Description
-name	n	false	.*	Existing assembly archive names.	Name of an assembly or assemblies, specified as a regular expression.

A.1.56.4 Examples

Here are some command examples.

A.1.56.4.1 Describe EM Assembly Archives

```
$ abctl describeEMAssemblyArchives [-name nameOfAssemblyArchive]
```

A.1.57 describeEMConnection

Details for this command follow.

A.1.57.1 Synopsis

```
$ abctl describeEMConnection
```

A.1.57.2 Description

Describes the configured EM Software Library connection.

A.1.57.3 Options

None.

A.1.57.4 Examples

Here are some command examples.

A.1.57.4.1 Describe EM Connection

```
$ abctl describeEMConnection
```

A.1.58 describeEndpoints

Details for this command follow.

A.1.58.1 Synopsis

```
$ abctl describeEndpoints -name String [-recurse]
```

A.1.58.2 Description

Lists the endpoints of an appliance or assembly. Specify -recurse to list endpoints of an assembly's sub-appliances and sub-assemblies.

A.1.58.3 Options

Table A–56 shows the command options for `describeEndpoints`.

Table A–56 describeEndpoints options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-name`	n	true	None.	Name of an appliance or assembly.	Name of an appliance or assembly whose endpoints will be listed.
`-recurse`	r	false	N/A	N/A	If specified, list the endpoints for an assembly's sub-elements instead of endpoints for the assembly itself.

A.1.58.4 Examples

Here are some command examples.

A.1.58.4.1 Describe Endpoints

```
$ abctl describeEndpoints -name mySite/myWls -r
```

A.1.59 describeFileSetDefinitions

Details for this command follow.

A.1.59.1 Synopsis

```
$ abctl describeFileSetDefinitions [-name] String
```

A.1.59.2 Description

Describes the file set definitions of an appliance or assembly.

A.1.59.3 Options

Table A–57 shows the command options for `describeFileSetDefinitions`.

Table A–57 describeFileSetDefinitions options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-name`	n	true	None.	Element name or path.	Name of an appliance or assembly whose definitions will be listed.

A.1.59.4 Examples

Here are some command examples.

A.1.59.4.1 Describe File Set Definitions

```
$ abctl describeFileSetDefinitions -name myOhs

$ abctl describeFileSetDefinitions -name /mySite/wls1/AdminServer
```

A.1.60 describeInterfaces

Details for this command follow.

A.1.60.1 Synopsis

```
$ abctl describeInterfaces -name String [-recurse]
```

A.1.60.2 Description

Lists the physical and virtual interfaces of an appliance or assembly.

A.1.60.3 Options

Table A–56 shows the command options for `describeInterfaces`.

Table A–58 describeInterfaces options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-name`	n	true	None.	Name of an appliance or assembly.	Name of an appliance or assembly whose interfaces will be listed.

A.1.60.4 Examples

Here are some command examples.

A.1.60.4.1 Describe Interfaces

```
$ abctl describeInterfaces -name mySite/ohs1

Interfaces in mySite/ohs1
--------------------------------------------------------------------------------
  Element      | Name            | Type    | Vnet   | Description
--------------------------------------------------------------------------------
/mySite/ohs1 | iface-1 (default)| Physical | vnet-1 | default appliance interface
/mySite/ohs1 | iface-1/vface-1  | Virtual  | vnet-1 | virtual interface for
iface-1
/mySite/ohs1 | iface-1/vface-2  | Virtual  | vnet-1 | another virtual interface
--------------------------------------------------------------------------------
```

A.1.61 describeLogEvents

Details for this command follow.

A.1.61.1 Synopsis

```
$ abctl describeLogEvents -connectionName String [-user String]
[-assemblyInstanceId String] [-afterTime String] [-beforeTime String]
```

A.1.61.2 Description

Describe log events based on specified input criteria. For time related parameters, the only supported formats are `yyyy-MM-dd'T'HH:mm:ss.SSS`, `yyyy-MMdd'T'HH:mm:ss` and `yyyy-MM-dd`.

A.1.61.3 Options

Table A–59 shows the command options for `describeLogEvents`.

Table A–59 describeLogEvents options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-connection Name`	c	true	none	A string representing the user name.	Specifies the user name to query upon.
`-user`	u	false	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
`-assemblyIn stanceId`	a	false	none	A string representing the `assemblyInstanceId`.	Specifies the identifier of the assembly instance to query upon.
`-afterTime`	at	false	none	A string representing the start time.	Specifies the start time of the log event query time frame.
`-beforeTime`	bt	false	none	A string representing the end time.	Specifies the end time of the log event query time frame.

A.1.61.4 Examples

Here are some command examples.

A.1.61.4.1 Describe Log Events

```
$ abctl describeLogEvents -user ovab -assemblyInstanceId MyId -afterTime
2012-03-01 -beforeTime 2012-03-19
```

A.1.62 describePlugins

Details for this command follow.

A.1.62.1 Synopsis

```
$ abctl describePlugins [-long]
```

A.1.62.2 Description

Lists the set of installed introspector plug-ins and extensions including their status and various other details.

A.1.62.3 Options

Table A–60 shows the command options for `describePlugins`.

Table A–60 describePlugins options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-long`	l	false	N/A	N/A	Lists all attributes of all plug-ins in long format (maximum detail).

A.1.62.4 Examples

Here are some command examples.

A.1.62.4.1 Describe Plug-ins

```
$ abctl describePlugins
```

A.1.63 describeRegistrations

Details for this command follow.

A.1.63.1 Synopsis

```
$ abctl describeRegistrations [-assembly String] [-version String] -connectionName
String
```

A.1.63.2 Description

This command describes one or more assembly registrations.

A.1.63.3 Options

Table A–61 shows the command options for `describeRegistrations`.

Table A–61 describeRegistrations options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-assembly`	a	false	none	A string representing the name of the assembly.	The name of an assembly.
`-version`	v	true	none	A string representing the version of the assembly.	Assembly version.
`-connection Name`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.63.4 Examples

Here are some command examples.

A.1.63.4.1 Describe Registrations

```
$ abctl describeRegistrations -assembly MyAssembly -version 1
```

A.1.64 describeRequests

Details for this command follow.

A.1.64.1 Synopsis

```
$ abctl describeRequests [-requestId String...] -connectionName String
```

A.1.64.2 Description

This command describes one or more previously issued asynchronous requests.

A.1.64.3 Options

Table A–62 shows the command options for `describeRequests`.

Table A–62 describeRequests options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-requestId`	r	false	none	A string representing the `requestId`.	The `requestId` of a previously issued asynchronous request.
`-connection Name`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.64.4 Examples

Here are some command examples.

A.1.64.4.1 Describe Requests

```
$ abctl describeRequests
```

A.1.65 describeResources

Details for this command follow.

A.1.65.1 Synopsis

```
$ abctl describeResources -tag String... -connectionName String
```

A.1.65.2 Description

Describe assembly archive resources based on specified tags.

A.1.65.3 Options

Table A–63 shows the command options for `describeResources`.

Table A–63 describeResources options

Name	Alias	Req'd	Default Values	Possible Values	Description
-tag	t	true	none	A set of tag names associated with assembly archive resources.	Specifies one or more tags with which the queried assembly archive resources are associated.
`-connection Name`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.65.4 Examples

Here are some command examples.

A.1.65.4.1 Describe Resources

```
$ abctl describeResources -tag tag1 tag2 -connectionName MyConnection
```

A.1.66 describeScalingGroups

Details for this command follow.

A.1.66.1 Synopsis

```
$ abctl describeScalingGroups [-assemblyInstanceId String...] [-scalingGroupId
String...] -connectionName String
```

A.1.66.2 Description

Describes one or more scaling groups.

A.1.66.3 Options

Table A–64 shows the command options for `describeScalingGroups`.

Table A–64 describeScalingGroups options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-assemblyInstanceId`	a	false	none	A string representing the `assemblyInstanceId`.	The identifier of a previously created assembly instance.
`-scalingGroupId`	s	false	none	A string representing the `scalingGroupId`.	The identifier of a previously created scaling group.
`-connectionName`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.66.4 Examples

Here are some command examples.

A.1.66.4.1 Describe Scaling Groups

```
$ abctl describeScalingGroups -c myDeployerConn
```

A.1.67 describeTags

Details for this command follow.

A.1.67.1 Synopsis

```
$ abctl describeTags -resource String [-tag String...] -connectionName String
```

A.1.67.2 Description

This command describes one or more tags associated with artifacts maintained by the Deployer.

A.1.67.3 Options

Table A–65 shows the command options for `describeTags`.

Table A–65 describeTags options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-resource`	-r	true	none	A string specifying the resource id of the resource.	Specifies the resource for which to get tag information.
`-tag`	-t	false	none	A string representing the name of the tag.	Specifies one or more tags for which to get the values.
`-connection Name`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.67.4 Examples

Here are some command examples.

A.1.67.4.1 Describe Tags

```
$ abctl describeTags -tag foo -resource MyResource
```

A.1.68 describeTargetConfigurations

Details for this command follow.

A.1.68.1 Synopsis

```
$ abctl describeTargetConfigurations [-target String...] -connectionName String
```

A.1.68.2 Description

This command describes one or more target configurations.

A.1.68.3 Options

Table A–66 shows the command options for `describeTargetConfigurations`.

Table A–66 describeDeployer options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-target`	t	false	none	A string representing the name of the target.	The name of one or more targets.
`-connection Name`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.68.4 Examples

Here are some command examples.

A.1.68.4.1 Describe Target Configurations

```
$ abctl describeTargetConfigurations -target MyTarget
```

A.1.69 describeTargetNames

Details for this command follow.

A.1.69.1 Synopsis

```
$ abctl describeTargetNames -type String -connectionName String
```

A.1.69.2 Description

Lists the names of the targets of the type specified.

A.1.69.3 Options

Table A–67 shows the command options for `describeTargetNames`.

Table A–67 describeTargetNames options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-type`	t	true	none	A string representing the type of target. Possible value is `ovm`.	The type of the target.
`-connection Name`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.69.4 Examples

Here are some command examples.

A.1.69.4.1 Describe Target Names

```
$ abctl describeTargetNames
```

A.1.70 describeTargetUsers

Details for this command follow.

A.1.70.1 Synopsis

```
$ abctl describeTargetUsers -target String -connectionName String
```

A.1.70.2 Description

Lists the users of the specified target.

A.1.70.3 Options

Table A–68 shows the command options for `describeTargetUsers`.

Table A–68 describeTargetUsers options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-target`	t	true	none	A string representing the name of the target.	The name of the target.
`-connection Name`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.70.4 Examples

Here are some command examples.

A.1.70.4.1 Describe Target Users

```
$ abctl describeTargetUsers -target MyTarget
```

A.1.71 describeTargets

Details for this command follow.

A.1.71.1 Synopsis

```
$ abctl describeTargets [-target String...] -connectionName String
```

A.1.71.2 Description

This command describes runtime information for one or more deployment targets.

A.1.71.3 Options

Table A–69 shows the command options for `describeTargets`.

Table A–69 describeTargets options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-target`	t	false	none	A string representing the name of the target.	The name of one or more targets.
`-connection Name`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.71.4 Examples

Here are some command examples.

A.1.71.4.1 Describe Targets

```
$ abctl describeTargets
```

A.1.72 describeUserTargets

Details for this command follow.

A.1.72.1 Synopsis

```
$ abctl describeUserTargets -user String -connectionName String
```

A.1.72.2 Description

This command describes one or more types of deployment targets.

A.1.72.3 Options

Table A–70 shows the command options for `describeUserTargets`.

Table A–70 describeUserTargets options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-user`	u	true	none	A string representing the username of the user.	The username of the user.
`-connection Name`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.72.4 Examples

Here are some command examples.

A.1.72.4.1 Describe User Targets

```
$ abctl describeUserTargets -user MyUser
```

A.1.73 describeVnets

Details for this command follow.

A.1.73.1 Synopsis

```
$ abctl describeVnets -target String [-id String...] -connectionName String
```

A.1.73.2 Description

This command describes one or more networks in the target environment.

A.1.73.3 Options

Table A–70 shows the command options for `describeVnets`.

Table A–71 describeVnets options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-connection Name`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
`-id`	i	false	none	A string representing the network ID.	The network IDs to describe.
`-target`	t	true	none	A string representing the target name.	The name of a target whose networks will be described.

A.1.73.4 Examples

Here are some command examples.

A.1.73.4.1 Describe Vnets

```
$ abctl describeVnets -target myTarget -c myDeployerConn
```

A.1.74 disablePlugin

Details for this command follow.

A.1.74.1 Synopsis

```
$ abctl disablePlugin -pluginName String
```

A.1.74.2 Description

Disables the specified plug-in or extension and recursively any child extensions.

A.1.74.3 Options

Table A–72 shows the command options for `disablePlugin`.

Table A–72 disablePlugin options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-pluginName`	pn	true	none	N/A	A single plug-in name.

A.1.74.4 Examples

Here are some command examples.

A.1.74.4.1 Disable Plug-in

```
$ abctl disablePlugin -pn WLS
```

A.1.75 downloadAssemblyArchive

Details for this command follow.

A.1.75.1 Synopsis

```
$ abctl downloadAssemblyArchive -name String -version String [-fileName Path]
-connectionName String
```

A.1.75.2 Description

This command downloads an assembly archive from the Deployer repository.

A.1.75.3 Options

Table A–73 shows the command options for `downloadAssemblyArchive`.

Table A–73 downloadAssemblyArchive options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-name`	n	true	none	A string representing the name of the assembly.	The name of the assembly archive.
`-version`	v	true	none	A string representing the version of the assembly.	The version of the assembly archive.
`-fileName`	r	true	none	A string representing the new name and/or location of the assembly.	The new name of the assembly archive.
`-connection Name`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.75.4 Examples

Here are some command examples.

A.1.75.4.1 Download Assembly Archive

```
$ abctl downloadAssemblyArchive -name MyAssembly -version 1 RenamedAssembly.ova
```

A.1.76 downloadAssemblyMetadata

Details for this command follow.

A.1.76.1 Synopsis

```
$ abctl downloadAssemblyMetadata -name String -version String [-fileName Path]
[-generatePlan] -connectionName String
```

A.1.76.2 Description

Downloads assembly metadata descriptor from the Deployer. This allows you to determine what is in the assembly without downloading the entire archive.

A.1.76.3 Options

Table A–74 shows the command options for `downloadAssemblyMetadata`.

Table A–74 downloadAssemblyMetadata options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-connectionName`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
`-fileName`	fn	false	none	An absolute or relative path to a file where the assembly metadata will be written.	The new name of the OVF. The file name where the assembly metadata will be written. If omitted, the metadata will be saved as <assembly name>.ovf in the current working directory. You may specify an absolute or relative path to a file.
`-generatePlan`	g	false	none	N/A	If set, generate a default deployment plan. The plan will be generated in the same location where the downloaded metadata is saved.
`-name`	n	true	none	A string representing the name of the assembly.	The name of the assembly for which a metadata descriptor will be downloaded.
`-version`	v	true	none	A string representing the version of the assembly.	Assembly version.

A.1.76.4 Examples

Here are some command examples.

A.1.76.4.1 Download Assembly Metadata

```
$ abctl downloadAssemblyMetadata -name MyAssembly -version 1
```

A.1.77 downloadAssemblyResources

Details for this command follow.

A.1.77.1 Synopsis

```
$ abctl downloadAssemblyResources -assemblyName String -version String
-connectionName String [-fileName Path]
```

A.1.77.2 Description

Downloads an assembly resources file from the Deployer repository.

A.1.77.3 Options

Table A–75 shows the command options for `downloadAssemblyResources`.

Table A–75 downloadAssemblyResources options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-assemblyName`	n	true	none	A string representing the name of the assembly.	The name of the assembly.

Table A–75 (Cont.) downloadAssemblyResources options

Name	Alias	Req'd	Default Values	Possible Values	Description
-version	v	true	none	A string representing the assembly version.	The assembly version.
-connectionName	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
-fileName	fn	false	none	An absolute or relative path to a file where the assembly archive is to be written.	The file name where the assembly archive is saved. If omitted, the assembly archive is saved as "<assemblyname>_resources_<version>.zip" in the archive directory. You may specify an absolute or relative path to a file.

A.1.77.4 Examples

Here are some command examples.

A.1.77.4.1 Download Assembly Resources File

```
$ abctl downloadAssemblyResources -assemblyName MyAssembly -version 1
-connectionName MyConnection
```

A.1.78 downloadDeploymentPlan

Details for this command follow.

A.1.78.1 Synopsis

```
$ abctl downloadDeploymentPlan -assemblyInstanceId String -connectionName String
[-fileName Path] [-display downloadDeploymentPlan] -assemblyName String -version
String -planName String -connectionName String [-fileName Path] [-display]
```

A.1.78.2 Description

Downloads a deployment plan from an existing deployment.

A.1.78.3 Options

Table A–76 shows the command options for `downloadDeploymentPlan`.

Table A–76 downloadDeploymentPlan options

Name	Alias	Req'd	Default Values	Possible Values	Description
-assemblyInstanceId	a	true	none	A string representing the assemblyInstanceId.	The identifier of an assembly instance.
-connectionName	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
-fileName	fn	false	none	An absolute or relative path to a file where the deployment plan is to be written.	The file name where the deployment plan is to be written. If omitted, the plan is saved as "Plan.xml" in the current working directory. You may specify an absolute or relative path to a file.
-display	ds	false	none	N/A	If set, show the deployment plan contents after download.

Table A–76 (Cont.) downloadDeploymentPlan options

Name	Alias	Req'd	Default Values	Possible Values	Description
-assemblyName	an	true	none	A string representing the name of the assembly.	The name of the assembly.
-version	v	true	none	A string representing the assembly version.	The assembly version.
-planName	pn	true	none	A string representing the name of the deployment plan.	The name of the deployment plan.

A.1.78.4 Examples

Here are some command examples.

A.1.78.4.1 Download Deployment Plan

```
$ abctl downloadDeploymentPlan -assemblyName FOO -version 1 -planName FOO
```

A.1.79 downloadEMAssemblyArchive

Details for this command follow.

A.1.79.1 Synopsis

```
$ abctl downloadEMAssemblyArchive -name String -version String [-downloadAs
String] [-force]
```

A.1.79.2 Description

Downloads an assembly archive from the Enterprise Manager Software Library and imports it into the Oracle Virtual Assembly Builder Studio catalog. The assembly archive is reverse engineered to have the Oracle Virtual Assembly Builder metadata, file sets and templates created and persisted in the catalog.

By default, the download fails if an assembly with the same name already exists in the catalog. The `-downloadAs` option can be used to download an assembly with a different name.

A.1.79.3 Options

Table A–77 shows the command options for `downloadEMAssemblyArchive`.

Table A–77 downloadEMAssemblyArchive options

Name	Alias	Req'd	Default Values	Possible Values	Description
-name	n	true	none	Assembly archive name.	Name of assembly archive to download.
-version	v	true	none	A string representing the version of the assembly archive.	Specifies the version of the assembly archive to download from Enterprise Manager Software Library.
-force	f	false	false	N/A	If set, overwrites an existing assembly in the catalog that has the same name as the imported assembly.
-downloadAs		false	none	Assembly archive name.	Name to assign to a downloaded assembly inside the catalog.

A.1.79.4 Examples

Here is a command example.

A.1.79.4.1 downloadEMAssemblyArchive

```
% abctl downloadEMAssemblyArchive -name archiveName -version 1.0 -force
-downloadAs newName
```

A.1.80 enablePlugin

Details for this command follow.

A.1.80.1 Synopsis

```
$ abctl enablePlugin [-pluginName] String [-recurse]
```

A.1.80.2 Description

Enables the specified introspector plug-in or extension.

A.1.80.3 Options

Table A–78 shows the command options for `enablePlugin`.

Table A–78 enablePlugin options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-pluginName`	pn	true	none	N/A	A single plug-in name.
`-recurse`	r	false	none	N/A	If specified, also enables all possible child extensions recursively.

A.1.80.4 Examples

Here are some command examples.

A.1.80.4.1 Enable Plug-in

```
$ abctl enablePlugin -pn WLS -recurse
```

A.1.81 encryptProperties

Details for this command follow.

A.1.81.1 Synopsis

```
$ abctl encryptProperties -propertyNames String... [-outputFile Path]
```

A.1.81.2 Description

Encrypts one or more values that are obtained by securely prompting without echoing. The encrypted values can either be printed to the screen or appended to a specified file. This command is intended for use when constructing property files used in conjunction with script-based facilities.

A.1.81.3 Options

Table A–79 shows the command options for `encryptProperties`.

Table A–79 encryptProperties options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-pluginName`	pn	false	none	A set of one or more space-separated property names.	Allows for the prompting of multiple values referenced by name.
`-outputFile`	of	false *	none	Path to a file that may or may not exist.	The output file where the specified properties and encrypted values should be written. If the file does not exist then it will be created. If the file exists then all new properties will be appended to the file and any pre-existing properties will be updated. May only be specified when specifying `propertyNames`.

A.1.81.4 Examples

Here are some command examples.

A.1.81.4.1 Encrypt Properties

```
$ abctl encryptProperties
```

A.1.82 encryptProperty

Details for this command follow.

A.1.82.1 Synopsis

```
$ abctl encrypt
```

A.1.82.2 Description

Prompts for a value and returns the encrypted form. The original value is not displayed to the screen.

A.1.82.3 Options

None.

A.1.82.4 Examples

Here are some command examples.

A.1.82.4.1 Encrypt Property

```
$ abctl encryptProperty
```

A.1.83 export

Details for this command follow.

A.1.83.1 Synopsis

```
$ abctl export -name String -toDir Path [-quiet] [-metadataOnly]
```

A.1.83.2 Description

Exports an appliance or assembly to disk so that it can later be imported to another catalog.

A.1.83.3 Options

Table A–80 shows the command options for `export`.

Table A–80 export options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-metadataOnly`	m	false	none	N/A	Indicates that only the metadata portion of the appliance or assembly will be exported.
`-name`	n	true	none	Top level appliance or assembly in the catalog. Nested appliances cannot be exported.	Name of a top level appliance or assembly in the catalog.
`-quiet`	q	false	none	N/A	By default, the command shows detailed progress/success messages. If `-quiet` is set, the command turns off verbose mode and shows only one or two progress/success messages.
`-toDir`	td	true	none	A path to a directory. The directory must be empty. A non-existing directory will be created.	Path to the directory to which a specified appliance or assembly will be exported. If a relative path is given, it will be relative to AB_INSTANCE.

A.1.83.4 Examples

Here are some command examples.

A.1.83.4.1 Regular export

```
$ abctl export -name mySite -toDir /tmp/mySite.export
Executing export to /tmp/mySite.export.
  Step 1 of 4: Copying from source to dest.
    Copying: 100% of   52MB completed.
    Copying: 100% of  690MB completed.
    Copying: 100% of   86MB completed.
    Copying: 100% of   405B completed.
    Copying: 100% of  188MB completed.
    Copying: 100% of  1024B completed.
  Step 2 of 4: Copying from source to dest completed.
  Step 3 of 4: Archiving temporary catalog.
    Zipping: 100% of  138MB completed.
    Copying: 100% of  690MB completed.
    Copying: 100% of  188MB completed.
  Step 4 of 4: Archiving temporary catalog completed.
Successfully exported to /tmp/mySite.export.
```

A.1.83.4.2 Export with -metadataOnly flag

```
$ abctl export -name mySite -toDir /tmp/mySite.export -metadataOnly
Executing export to /tmp/mySite.export.
  Step 1 of 2: Archiving temporary catalog.
    Zipping: 100% of  6163B completed.
  Step 2 of 2: Archiving temporary catalog completed.
Successfully exported to /tmp/mySite.export.
```

A.1.83.4.3 Export to a non-empty directory

```
$ abctl export -name myOhs -toDir /tmp/non-empty-dir
Executing export to /tmp/non-empty-dir.
Error: OAB-7443: Failed to export myWls to /tmp/non-empty-dir.
Caused by: OAB-09509: Directory is not empty at /tmp/non-empty-dir.
  Action: Clean up the directory, or choose an empty directory.
```

A.1.84 findPlugins

Details for this command follow.

A.1.84.1 Synopsis

```
$ abctl findPlugins -productRoot Path [-remoteHost String] [-remoteUser String]
[-privateKeyFile Path] [-long]
```

A.1.84.2 Description

Lists the set of introspector plug-ins and extensions from a product home that you can install into OVAB.

A.1.84.3 Passwords

You may be prompted to enter a value for the password for the remote SSH user when you execute the command.

A.1.84.4 Options

Table A–81 shows the command options for `findPlugins`.

Table A–81 findPlugins options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-productRoot`	pr	true	none	Path.	Full path to the base directory of the product installation. For Oracle product installations, this can be either an ORACLE_HOME or one directory below an ORACLE_HOME to search multiple ORACLE_HOME.
`-remoteHost`	rh	false	none	N/A	Host name or IP address and optional SSH port of the remote machine. If set, the remoteUser must be specified as well.

Table A–81 (Cont.) findPlugins options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-remoteUser`	ru	false	none	N/A	Name of the ssh user to use for accessing the remote machine. If set, the remoteHost must be specified as well.
`-privateKeyFile`	pkf	false	none	N/A	Private SSH key file on the local machine.
`-long`	l	false	N/A	N/A	Lists all attributes of all plug-ins in long format (maximum detail).

A.1.84.5 Examples

Here are some command examples.

A.1.84.5.1 Find Plug-ins

```
$ abctl findPlugins
```

A.1.85 getCatalogProperty

Details for this command follow.

A.1.85.1 Synopsis

```
$ abctl getCatalogProperty -containerAddress String [-name String]
```

A.1.85.2 Description

This command enable getting certain property values within the catalog metadata. Not all properties in the catalog are gettable, and only some of these are settable (see setCatalogProperty).

A.1.85.3 Options

Table A–82 shows the command options for `getCatalogProperty`.

Table A–82 getCatalogProperty options

Name	Alias	Req'd	Default Values	Possible Values	Description
-containerAddress	c	true	none	A variable number of arguments.	Takes a variable number of arguments to specify a single container (such as an assembly, appliance, vnet, or physical interface) within the catalog's metadata hierarchy. It comprises a series of 'steps' one would take for navigating down through this hierarchy, starting from the root (catalog) level. The argument follows a Unix-like syntax although it does not refer to an actual directory path. It always begin with a slash ("/") character. For instance, to specify the Resource Requirements of the AdminServer appliance of a Weblogic server named wls1 of an assembly named "mySite", the container address is: /assemblies/mySite/assemblies/wls1/appliances/AdminServer/resource-requirements
-name	c	false	none	A string representing the name of a single property.	Specifies a single property in the output. This is useful primarily to enable a script to extract (scrape) a specific value from stdout. If the name parameter is omitted, all properties and child containers of the specified container are displayed.

A.1.85.4 Examples

Here are some command examples.

A.1.85.4.1 Get Catalog Property (all properties and child containers of the specified container)

```
$ abctl getCatalogProperty -containerAddress /assemblies/mySite

 dr-  assemblies
  dr-  appliances
  dr-  vnets

  -r-  assembly-id: LxVX9hK8CpZYW
  -r-  atomic: false
  -r-  capture-id: KxVX9hK8CpZYW
  -r-  default-network-name: vnet-1
  -rw  dependence:
  -rw  description: mySite
  -r-  name: mySite
  -r-  version: 1.0.0
```

The codes listed at the start of each line are useful for understanding options available for subsequent commands:

```
 dr-   denotes a child container. You can append this 'directory' to the current
container address, which affects navigating down to another container level.

 -r-   denotes a read-only property. You cannot update this property with the
setCatalogProperty command.
```

```
 -rw   denotes a writeable property. You can update this property through the
setCatalogProperty command.
```

Based on the output of the previous usage example, the following commands are expected to be valid:

```
$ abctl getCatalogProperty -containerAddress /assemblies/mySite/assemblies
$ abctl getCatalogProperty -containerAddress /assemblies/mySite/appliances
$ abctl getCatalogProperty -containerAddress /assemblies/mySite/vnets

$ abctl getCatalogProperty -containerAddress /assemblies/mySite -name assembly-id
$ abctl getCatalogProperty -containerAddress /assemblies/mySite -name atomic
$ abctl getCatalogProperty -containerAddress /assemblies/mySite -name capture-id
$ abctl getCatalogProperty -containerAddress /assemblies/mySite -name
default-network-name
$ abctl getCatalogProperty -containerAddress /assemblies/mySite -name dependence
$ abctl getCatalogProperty -containerAddress /assemblies/mySite -name description
$ abctl getCatalogProperty -containerAddress /assemblies/mySite -name name
$ abctl getCatalogProperty -containerAddress /assemblies/mySite -name version
$ abctl setCatalogProperty -containerAddress /assemblies/mySite -name dependence
-value ""
$ abctl setCatalogProperty -containerAddress /assemblies/mySite -name description
-value "New description value"
```

A.1.86 getDefaultTarget

Details for this command follow.

A.1.86.1 Synopsis

```
$ abctl getDefaultTarget -connectionName String
```

A.1.86.2 Description

This command returns the default target.

A.1.86.3 Options

Table A–83 shows the command options for `getDefaultTarget`.

Table A–83 getDefaultTarget options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-connection Name`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.86.4 Examples

Here are some command examples.

A.1.86.4.1 Get Default Target

```
$ abctl getDefaultTarget -c myDeployerConn
```

A.1.87 getTargetType

Details for this command follow.

A.1.87.1 Synopsis

```
$ abctl getTargetType -name String -connectionName String
```

A.1.87.2 Description

This command returns the type of the target.

A.1.87.3 Options

Table A–84 shows the command options for `getTargetType`.

Table A–84 getTargetType options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-name`	n	true	none	A string representing the name of the target	The name of the target.
`-connection Name`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.87.4 Examples

Here are some command examples.

A.1.87.4.1 Get Target Type

```
$ abctl getTargetType -name MyTarget
```

A.1.88 help

Details for this command follow.

A.1.88.1 Synopsis

```
help [[-command] string] [-usage]
```

A.1.88.2 Description

Prints a brief help message or more detailed help for a specified command.

A.1.88.3 Options

Table A–85 shows the command options for `help`.

Table A–85 help options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-command`	c	false	none	Any command of this utility.	Specifies the command for which Help should be printed.
`-usage`	u	false	none	N/A	Print only an option summary of the specified command.

A.1.88.4 Examples

Here is an example.

A.1.88.4.1 Print help, help for introspectWLS command and option summary for import command

```
abctl help, abctl help -command introspectWLS,
abctl help -usage -command import
```

A.1.89 import

Details for this command follow.

A.1.89.1 Synopsis

```
$ abctl import -from Path [-quiet] [-importAs String] [-force]
```

A.1.89.2 Description

Imports an appliance from a specified directory or an assembly from either a specified directory or a specified assembly archive.

A.1.89.3 Options

Table A–85 shows the command options for `import`.

Table A–86 import options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-from`	fr	true	none	An assembly archive file or a directory containing an exported appliance or assembly.	Path to a valid export location or assembly archive file.
`-quiet`	q	false	none	N/A	By default, the command shows detailed progress/success messages. If `-quiet` is set, the command turns off verbose mode and shows only one or two progress/success messages.
`-importAs`	ia	false	none	A unique name among top-level appliances or assemblies in a catalog.	If specified, imported appliancie or assembly will be saved with the given name in the catalog.
`-force`	f	false	false	N/A	If specified, existing top-level appliance or assembly in catalog using the same name as imported appliance or assembly will be overwritten.

A.1.89.4 Examples

Here are some command examples.

A.1.89.4.1 Import Assembly Archive

```
$ abctl import -from /tmp/mySite.ova -importAs myNewSite
```

A.1.89.4.2 Import from an Export Location

```
$ abctl import -fromDir /tmp/myWls.export -importAs wls_1
```

A.1.90 installPlugins

Details for this command follow.

A.1.90.1 Synopsis

```
$ abctl installPlugins -productRoot Path [-pluginNames String...] [-remoteHost
String] [-remoteUser String] [-privateKeyFile Path]
```

A.1.90.2 Description

Installs one or more plug-ins and extensions from a product home.

A.1.90.3 Passwords

You may be prompted to enter a value for the password for the remote SSH user when you execute the command.

A.1.90.4 Options

Table A–87 shows the command options for `installPlugins`.

Table A–87 installPlugins options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-pluginNames`	pn	true	none	N/A	One or more plug-in names separated by spaces.
`-productRoot`	pr	true	none	Path.	Full path to the base directory of the product installation. For Oracle product installations, this can be either an ORACLE_HOME or one directory below an ORACLE_HOME to search multiple ORACLE_HOME.
`-remoteHost`	rh	false	none	N/A	Host name or IP address and optional SSH port of the remote machine. If set, the remoteUser must be specified as well.
`-remoteUser`	ru	false	none	N/A	Name of the ssh user to use for accessing the remote machine. If set, the remoteHost must be specified as well.
`-privateKeyFile`	pkf	false	none	N/A	Private SSH key file on the local machine.

A.1.90.5 Examples

Here are some command examples.

A.1.90.5.1 Install Plug-ins

```
$ abctl installPlugins
```

A.1.91 introspectCoherenceWeb

Details for this command follow.

A.1.91.1 Synopsis

```
$ abctl introspectCoherenceWeb -wlsHome Path -domainRoot Path -adminUser String
[-name String] [-force] [-skipFileSetCapture] [-remoteHost String] [-remoteUser
String] [-sudoUser String] [-remoteWorkingDir Path] [-remoteCleanup]
[-privateKeyFile Path] [-description]
```

A.1.91.2 Description

This command is an alias for `introspectWLS`. It examines the configuration of an installed WebLogic domain to determine what file sets must be captured and what

configuration must be changed at deployment. All collected data is stored in the catalog upon successful completion.

A.1.91.3 Options

Table A–88 shows the command options for `introspectCoherenceWeb`.

Table A–88 introspectCoherenceWeb options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-adminUser`	au	true	none	N/A	Administrative name for the WebLogic domain.
`-description`	d	false	none	N/A	Description of the appliance or assembly being introspected.
`-domainRoot`	dr	true	none	N/A	Full path to the WebLogic domain root.
`-force`	f	false	none	N/A	Overwrite any introspection in the catalog that exists with the same name.
`-name`	n	false	Derived directory name prefixed by component type name.	Any name not already used within the catalog.	Specifies a name by which the introspection output is stored.
`-privateKeyFile`	pkf	false	none	N/A	Private SSH key file on the local machine.
`-remoteCleanup`	rc	false	none	N/A	Remote clean up flag. When set, the remote working directory will be deleted after the operation. Otherwise the directory will not be modified. If set, the remoteUser and remoteHost must be specified as well.
`-remoteHost`	rh	false	none	N/A	Host name or IP address and optional SSH port of the remote machine. If set, the remoteUser must be specified as well.
`-remoteUser`	ru	false	none	N/A	Name of the ssh user to use for accessing the remote machine. If set, the remoteHost must be specified as well.
`-remoteWorkingDir`	rwd	false	`/tmp/abRemote_<remote user name>`	N/A	Path on the remote machine to work out of. If set, the remoteUser and remoteHost must be specified as well.
`-skipFileSetCapture`	sf	false	none	N/A	If specified, file sets are not captured for the component during introspection.
`-sudoUser`	su	false	none	User name of sudo user.	Specifies a sudo user. When specified, remote capturing of file sets or introspection will substitute the user (sudo) as the sudo user before running the remote Assembly Builder. If sudoUser is specified, you cannot use the privateKeyFile. That is, sudoUser can only be used when you provide a password.
`-wlsHome`	wh	true	none	N/A	Full path to the WebLogic home directory, generally `<middleware home>/wlserver`.

A.1.91.4 Examples

Here is a command example.

A.1.91.4.1 Basic Introspection of a Coherence Appliance

This is a basic introspection of a Coherence appliance to a specific catalog, using a capture name of myIntrospection.

```
$ abctl introspectCoherenceWeb -name myIntrospection <Coherence options>
```

A.1.92 introspectForms

Details for this command follow.

A.1.92.1 Synopsis

```
introspectForms -wlsHome Path -domainRoot Path -adminUser String [-soaGlobalCP
Path] [-name String] [-force] [-skipFileSetCapture] [-remoteHost String]
[-remoteUser String] [-sudoUser String] [-remoteWorkingDir Path] [-remoteCleanup]
[-privateKeyFile Path] [-description]
```

A.1.92.2 Description

This command is an alias for `introspectWLS`. Examines the configuration of an installed Oracle WebLogic Server domain to determine what file sets needs to be captured and what configuration needs to be changed at deployment.

All collected data is stored in the catalog upon successful completion.

A.1.92.3 Options

Table A–89 shows the command options for `introspectForms`.

Table A–89 introspectForms options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-adminUser`	au	true	none	N/A	Administrative name for the Oracle WebLogic Server domain.
`-description`	d	false	none	N/A	Description of the appliance or assembly being introspected.
`-domainRoot`	dr	true	none	N/A	Full path to the Oracle WebLogic Server domain root.
`-force`	f	false	none	N/A	Overwrite any introspection in the catalog that exists with the same name.
`-name`	n	false	Derived directory name prefixed by component type name.	Any name not already used within the catalog.	Specifies a name by which the introspection output is stored.
`-privateKeyFile`	pkf	false	none	N/A	Private SSH key file on the local machine.
`-remoteCleanup`	rc	false	false	N/A	Remote clean up flag. When set, the remote working directory will be deleted after the operation. Otherwise the directory will not be modified. If set, `remoteUser` and `remoteHost` must be specified as well.
`-remoteHost`	rh	false	none	N/A	Host name or IP address and optional SSH port of the remote machine. If set, `remoteUser` must be specified as well.
`-remoteUser`	ru	false	none	N/A	Name of the SSH user to use for accessing the remote machine. If set, `remoteHost` must be specified as well.
`-remoteWorkingDir`	rwd	false	`/tmp/abRemote_<remote user name>`	N/A	Path on the remote machine to work out of. If set, `remoteUser` and `remoteHost` must be specified as well.
`-soaGlobalCP`	sgcp	false	none	Location of optional global configuration plan.	The absolute path to an optional global configuration plan.

Table A–89 (Cont.) introspectForms options

Name	Alias	Req'd	Default Values	Possible Values	Description
-skipFileSetCapture	sf	false	none	N/A	If specified, file sets are not captured for the component during introspection.
-sudoUser	su	false	none	User name of sudo user.	Specifies a sudo user. When specified, remote capturing of file sets or introspection will substitute the user (sudo) as the sudo user before running the remote Assembly Builder. If sudoUser is specified, you cannot use the privateKeyFile. That is, sudoUser can only be used when you provide a password.
-wlsHome	wh	true	none	WLS Home	Full path to the Oracle WebLogic Server home (usually, <middleware home>/wlserver_10.3).

A.1.92.4 Examples

Here are some command examples.

A.1.92.4.1 Basic Introspection

This is a basic introspection of a Forms appliance.

```
abctl introspectForms <Forms options>
```

A.1.92.4.2 Introspection with a specific capture name

An introspection of component "Forms" saved with an appliance/assembly name of "myIntrospection":

```
% abctl introspectForms -name myIntrospection <Forms options>
```

A.1.93 introspectGenericProd

Details for this command follow.

A.1.93.1 Synopsis

```
$ abctl introspectGenericProd -productRoots String [-propertyFile Path]
[-scriptRootDir Path] [-name String] [-force] [-skipFileSetCapture] [-remoteHost
String] [-remoteUser String] [-sudoUser String] [-remoteWorkingDir Path]
[-remoteCleanup] [-privateKeyFile Path] [-description]
```

A.1.93.2 Description

Captures products generically by taking as input the set of product directories to capture, a set of properties that can be modified by the user, and a set of scripts to run on the appliance instance during deployment operations.

A.1.93.3 Options

Table A–90 shows the command options for introspectGenericProd.

Table A–90 introspectGenericProd options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-description`	d	false	none	N/A	Description of the appliance or assembly being introspected.
`-force`	f	false	none	N/A	Overwrite any introspection in the catalog that exists with the same name.
`-name`	n	false	Derived directory name prefixed by component type name.	Any name not already used within the catalog.	Specifies a name by which the introspection output is stored.
`-privateKeyFile`	pkf	false	none	N/A	Private SSH key file on the local machine.
`-productRoots`	pr	true	none	The colon-separated list of product directories to capture.	A list of one or more colon-separated paths. Each path must be a directory that exists. All files and directories within each specified directory are captured as file sets. All specified paths are available at the same locations on the appliance instance during deployment.

Table A–90 (Cont.) introspectGenericProd options

Name	Alias	Req'd	Default Values	Possible Values	Description
-propertyFile	pf	true	none	Properties file containing properties to add to the appliance.	If the propertyFile parameter is specified then it must point to a file that exists and is readable. A property file must be a text file containing a list of name/value pairs. Each property in the property file will be added as a user property into the appliance. Like any other user property, these properties can be edited in the generated appliance and can be overridden in deployment plans. During deployment the properties will be written back out to a file, including the values as edited within the appliance or overridden in the deployment plan. The regenerated properties file will be made available to all scripts during their execution through an environment variable named '$AB_USERPROPS_FILE'. These properties are intended for eventual consumption by the scripts captured through the 'scriptRootDir' parameter. For this reason, the property names and values must be in a format that can be sourced by a shell script. Each line in a property file must consist of zero or more lines where each line must be a property declaration, a comment, or a blank line. Each property declaration must be equivalent to a variable declaration (name=value) where the name can be converted to an environment variable. Property declarations must be contained on a single line. Ending a line with '\' will not result in line continuation. Comments and blank lines are discarded at dehydration and will not be reproduced when the file is regenerated at reconfiguration. All properties will be marked as 'required' in the appliance metadata. Property declarations without any assigned value (nothing after '=') will be set to null in the appliance metadata, requiring that the user assign a value to that property prior to deployment. Whitespace is not permitted anywhere to the left of '=' in a property declaration. Whitespace to the right of '=' is assumed to be part of the intended value and will be preserved (resulting in failure if the value is sourced). Quotes around property values will be preserved and will be visible to scripts as part of the value. When editing a property value, it is the responsibility of the user to add/remove/preserve quotes as necessary according to the rules of shell interpretation.
-remoteCleanup	rc	false	none	N/A	Remote clean up flag. When set, the remote working directory will be deleted after the operation. Otherwise the directory will not be modified. If set, the remoteUser and remoteHost must be specified as well.
-remoteHost	rh	false	none	N/A	Host name or IP address and optional SSH port of the remote machine. If set, the remoteUser must be specified as well.

Table A–90 (Cont.) introspectGenericProd options

Name	Alias	Req'd	Default Values	Possible Values	Description
-remoteUser	ru	false	none	N/A	Name of the ssh user to use for accessing the remote machine. If set, the remoteHost must be specified as well.
-remoteWorkingDir	rwd	false	/tmp/abRemote_<remote user name>	N/A	Path on the remote machine to work out of. If set, the remoteUser and remoteHost must be specified as well.
-scriptRootDir	srd	false	none	The top level directory containing the script subdirectories.	The script root directory is the top level directory containing the script subdirectories. If the specified directory does not exist or is not readable then an error will be returned and an appliance will not be created. Scripts must be placed within the root script directory under the following well-known subdirectories: config.d/, start.d/, ping.d/, stop.d/. Scripts under each subdirectory will be captured during introspection and stored with the appliance. During deployment the appropriate set of scripts according to the requested operation will be executed sequentially. The script root directory need not contain all well-known subdirectories and well-known subdirectories that do exist may be empty. All scripts are executed as the root user to provide the flexibility of performing operations requiring root privileges or switching to another user as necessary. The path to a properties file containing the variables specified at introspection will be made available during script execution through the '$AB_USERPROPS_FILE' environment variable. This file can be sourced by the script. All scripts must exit with a zero exit status upon success. Any script exiting with a non-zero exit status will result in the failure of the operation.
-skipFileSetCapture	sf	false	none	N/A	If specified, file sets are not captured for the component during introspection.
-sudoUser	su	false	none	User name of sudo user.	Specifies a sudo user. When specified, remote capturing of file sets or introspection will substitute the user (sudo) as the sudo user before running the remote Assembly Builder. If sudoUser is specified, you cannot use the privateKeyFile. That is, sudoUser can only be used when you provide a password.

A.1.93.4 Examples

Here is a command example.

A.1.93.4.1 Basic Introspection of Appliance "GenericProd"

This is a basic introspection of appliance Coherence to a specific catalog, using a capture name of myIntrospection.

```
$ abctl introspectGenericProd <GenericProd options>
```

A.1.93.4.2 Basic Introspection of Appliance "GenericProd"

This is an introspection of a generic appliance saved with an appliance/assembly name of "myIntrospection".

```
$ abctl introspectGenericProd -name myIntrospection <GenericProd options>
```

A.1.94 introspectOHS

Details for this command follow.

A.1.94.1 Synopsis

```
introspectOHS -oracleInstance Path -componentName String [-name string] [-force]
[-skipFileSetCapture] [-remoteHost String] [-remoteUser String] [-sudoUser String]
[-remoteWorkingDir Path] [-remoteCleanup] [-privateKeyFile Path] [-description]
```

A.1.94.2 Description

Examines the configuration of an installed OHS component to determine what file sets must be captured and what configuration must be changed at deployment. All collected data is stored in the catalog upon successful completion.

A.1.94.3 Options

Table A–91 shows the command options for `introspectOHS`.

Table A–91 introspectOHS options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-componentName`	cn	true	none	N/A	The name of the Oracle HTTP Server appliance to introspect (for example: `ohs1`).
`-description`	d	false	none	N/A	Description of the appliance or assembly being introspected.
`-force`	f	false	none	N/A	Overwrite any introspection in the catalog that exists with the same name.
`-name`	n	false	Derived directory name prefixed by component type name	Any name not previously used within the catalog	Specifies a name by which the introspection output is stored.
`-oracleInstance`	oi	true	none	N/A	The absolute path of the ORACLE_INSTANCE to introspect.
`-privateKeyFile`	pkf	false	none	Location of a private key file.	Private SSH key file on the local machine.
`-remoteCleanup`	rc	false	false	N/A	Remote clean up flag. When set, the remote working directory will be deleted after the operation. Otherwise the directory will not be modified. If set, `remoteUser` and `remoteHost` must be specified as well.
`-remoteHost`	rh	false	none	N/A	Host name or IP address and optional SSH port of the remote machine. If set, `remoteUser` must be specified as well.
`-remoteUser`	ru	false	none	N/A	Name of the SSH user to use for accessing the remote machine. If set, `remoteHost` must be specified as well.

Table A–91 (Cont.) introspectOHS options

Name	Alias	Req'd	Default Values	Possible Values	Description
-remoteWorkingDir	rwd	false	/tmp/abRemote	N/A	Path on the remote machine to work out of. If set, remoteUser and remoteHost must be specified as well.
-skipFileSetCapture	sf	false	none	N/A	If specified, file sets are not captured for the component during introspection.
-sudoUser	su	false	none	User name of sudo user.	Name of the user on the remote machine to sudo as before executing operations. If sudoUser is specified, you cannot use the privateKeyFile. That is, sudoUser can only be used when you provide a password.

A.1.94.4 Examples

Here are some command examples.

A.1.94.4.1 Successful Introspection

```
% abctl introspectOHS -name myOHS -oracleInstance /ora/inst1 -componentName ohs1
Launching introspection of appliance 'OHS' ...
  Step 1 of 5: OHS introspection starting
    Step 1 of 4: OHS Httpd Configuration parsed
    Step 2 of 4: OHS Httpd configuration transformed
    Step 3 of 4: OHS Httpd configuration processed
    Step 4 of 4: OHS Httpd configuration written
  Step 2 of 5: HTTPD processing completed
    Step 1 of 3: OHS OPMN configuration parsed
    Step 2 of 3: OHS OPMN configuration processed
    Step 3 of 3: OHS OPMN configuration writtend
  Step 3 of 5: OPMN XML processing completed
    Step 1 of 2: OHS opmnctl script parsed
    Step 2 of 2: Appliance updated with ORACLE_HOME
  Step 4 of 5: OPMNCTL processing completed
  Step 5 of 5: OHS introspection complete
Task is done: DehydrateJob completed
Introspection complete
Storing result in catalog: '/Oracle/IntrospectionCatalog' ...
Introspection stored as 'myohs' in the catalog
%
```

A.1.94.4.2 Failed Introspection bad -oracleInstance value

```
% abctl introspectOHS -oracleInstance /ora/dontexist -componentName foobar
Launching introspection of appliance 'OHS' ...
  Step 1 of 5: OHS task starting
Task is done: DehydrateJob failed with error: The specified Oracle Instance does
not exist.
Error: Introspection failed
Caused by: The specified Oracle Instance does not exist.
%
```

A.1.95 introspectOTD

Details for this command follow.

A.1.95.1 Synopsis

```
introspectOTD -oracleHome Path -oracleInstance Path -configName String [-name
```

```
String] [-force] [-skipFileSetCapture] [-remoteHost String] [-remoteUser String]
[-sudoUser String] [-remoteWorkingDir Path] [-remoteCleanup] [-privateKeyFile
Path] [-description]
```

A.1.95.2 Description

Examines the configuration of an installed Oracle Traffic Director configuration to determine what file sets need to be captured and what configuration needs to be changed at deployment. All collected data is stored in the catalog upon successful completion. Note that Oracle Traffic Director administration server will not be introspected and will be recreated from scratch during reconfiguration.

A.1.95.3 Options

Table A–92 shows the command options for `introspectOTD`.

Table A–92 introspectOTD options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-configName`	cn	true	none	Name of the Oracle Traffic Director configuration.	Specifies the name of an Oracle Traffic Director configuration which needs to be introspected.
`-description`	d	false	none	N/A	Description of the appliance or assembly being introspected.
`-force`	f	false	none	N/A	Overwrite any introspection in the catalog that exists with the same name.
`-name`	n	false	Derived directory name prefixed by component type name	Any name not previously used within the catalog	Specifies a name by which the introspection output is stored.
`-oracleHome`	oh	true	none	The directory where Oracle Traffic Director is installed.	Specify the absolute path to the directory where Oracle Traffic Director is installed.
`-oracleInstance`	oi	true	none	N/A	The absolute path of the ORACLE_INSTANCE to introspect.
`-privateKeyFile`	pkf	false	none	Location of a private key file.	Private SSH key file on the local machine.
`-remoteCleanup`	rc	false	false	N/A	Remote clean up flag. When set, the remote working directory will be deleted after the operation. Otherwise the directory will not be modified. If set, `remoteUser` and `remoteHost` must be specified as well.
`-remoteHost`	rh	false	none	N/A	Host name or IP address and optional SSH port of the remote machine. If set, `remoteUser` must be specified as well.
`-remoteUser`	ru	false	none	N/A	Name of the SSH user to use for accessing the remote machine. If set, `remoteHost` must be specified as well.

Table A–92 (Cont.) introspectOTD options

Name	Alias	Req'd	Default Values	Possible Values	Description
-remoteWorkingDir	rwd	false	/tmp/abRemote	N/A	Path on the remote machine to work out of. If set, remoteUser and remoteHost must be specified as well.
-skipFileSetCapture	sf	false	none	N/A	If specified, file sets are not captured for the component during introspection.
-sudoUser	su	false	none	User name of sudo user.	Name of the user on the remote machine to sudo as before executing operations. If sudoUser is specified, you cannot use the privateKeyFile. That is, sudoUser can only be used when you provide a password.

A.1.95.4 Examples

Here are some command examples.

A.1.95.4.1 Basic Introspection

```
% abctl introspectOTD <OTD options>
```

A.1.95.4.2 Introspection of appliance "OTD" saved with the name "myIntrospection"

```
% abctl introspectOTD -name myIntrospection <OTD options>
```

A.1.96 introspectRACDB

Details for this command follow.

A.1.96.1 Synopsis

```
introspectRACDB -crsHome Path -dbHome Path [-globalDbName String] [-sysDBAUserName
String] -shutdownDBOK String [-asmHome Path] [-name String] [-force]
[-skipFileSetCapture] [-remoteHost String] [-remoteUser String] [-sudoUser String]
[-remoteWorkingDir Path] [-remoteCleanup] [-privateKeyFile Path] [-description]
```

A.1.96.2 Description

Examines CRS and RAC Database configuration and captures metadata.

A.1.96.3 Options

Table A–93 shows the command options for introspectRACDB.

Table A–93 introspectRACDB options

Name	Alias	Req'd	Default Values	Possible Values	Description
-asmHome	ch	false	none	N/A	This parameter is required if ASM is used as the storage type and it is installed in a separate Oracle Home.
-crsHome	ch	true	none	N/A	The ORACLE_HOME of the Oracle CRS to be introspected.
-dbHome	dh	true	none	N/A	The ORACLE_HOME of the Oracle RDBMS to be introspected.
-description	d	false	none	N/A	Description of the appliance or assembly being introspected.

Table A–93 (Cont.) introspectRACDB options

Name	Alias	Req'd	Default Values	Possible Values	Description
-globalDbName	dun	false	value specified for -oracleSid	N/A	The global database name of the Oracle RDBMS to be introspected.
-force	f	false	none	N/A	Overwrite any introspection in the catalog that exists with the same name.
-name	n	false	Derived directory name prefixed by component type name.	Any name not already used within the catalog.	Specifies a name by which the introspection output is stored
-privateKeyFile	pkf	false	none	N/A	Private SSH key file on the local machine.
-remoteCleanup	rc	false	none	N/A	Remote clean up flag. When set, the remote working directory will be deleted after the operation. Otherwise the directory will not be modified. If set, `remoteUser` and `remoteHost` must be specified as well.
-remoteHost	rh	false	none	N/A	Host name or IP address and optional SSH port of the remote machine. If set, `remoteUser` must be specified as well.
-remoteUser	ru	false	none	N/A	Name of the ssh user to use for accessing the remote machine. If set, `remoteHost` must be specified as well.
-remoteWorkingDir	rwd	false	/tmp/abRemote_<remote user name>	N/A	Path on the remote machine to work out of. If set, `remoteUser` and `remoteHost` must be specified as well.
-shutdownDBOK	sdbok	true	none	N/A	This flag needs to be passed to approve the database reboot.
-skipFileSetCapture	sf	false	none	N/A	If specified, file sets are not captured for the component during introspection.
-sysDBAUserName	sdbaun	false	none	N/A	Database account with SYSDBA privileges. This parameter is required only if OS authentication is disabled for the current database.
-sudoUser	su	false	none	User name of sudo user.	Specifies a sudo user. When specified, remote capture of file sets or introspection will substitute the user (sudo) as the sudo user before running the remote Assembly Builder. If sudoUser is specified, you cannot use the privateKeyFile. That is, sudoUser can only be used when you provide a password.

A.1.96.4 Examples

Here are some command examples.

A.1.96.4.1 Basic Introspection

This is a basic introspection of a single-instance DB appliance.

```
abctl introspectRACDB <DB options>
```

A.1.96.4.2 Introspection into a specific catalog with a specific capture name

```
% abctl introspectRACDB -name myIntrospection <DB options>
```

A.1.97 introspectReports

Details for this command follow.

A.1.97.1 Synopsis

```
introspectReports -wlsHome Path -domainRoot Path -adminUser String [-soaGlobalCP
Path] [-name String] [-force] [-skipFileSetCapture] [-remoteHost String]
[-remoteUser String] [-sudoUser String] [-remoteWorkingDir Path] [-remoteCleanup]
[-privateKeyFile Path] [-description]
```

A.1.97.2 Description

This command is an alias for `introspectWLS`. Examines the configuration of an installed Oracle WebLogic Server domain to determine what file sets needs to be captured and what configuration needs to be changed at deployment.

All collected data is stored in the catalog upon successful completion.

A.1.97.3 Options

Table A–96 shows the command options for `introspectReports`.

Table A–94 introspectReports options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-adminUser`	au	true	none	N/A	Administrative name for the Oracle WebLogic Server domain.
`-description`	d	false	none	N/A	Description of the appliance or assembly being introspected.
`-domainRoot`	dr	true	none	N/A	Full path to the Oracle WebLogic Server domain root.
`-force`	f	false	none	N/A	Overwrite any introspection in the catalog that exists with the same name.
`-name`	n	false	Derived directory name prefixed by component type name.	Any name not already used within the catalog.	Specifies a name by which the introspection output is stored.
`-privateKeyFile`	pkf	false	none	N/A	Private SSH key file on the local machine.
`-remoteCleanup`	rc	false	false	N/A	Remote clean up flag. When set, the remote working directory will be deleted after the operation. Otherwise the directory will not be modified. If set, `remoteUser` and `remoteHost` must be specified as well.
`-remoteHost`	rh	false	none	N/A	Host name or IP address and optional SSH port of the remote machine. If set, `remoteUser` must be specified as well.

Table A–94 (Cont.) introspectReports options

Name	Alias	Req'd	Default Values	Possible Values	Description
-remoteUser	ru	false	none	N/A	Name of the SSH user to use for accessing the remote machine. If set, remoteHost must be specified as well.
-remoteWorkingDir	rwd	false	/tmp/abRemote_<remote user name>	N/A	Path on the remote machine to work out of. If set, remoteUser and remoteHost must be specified as well.
-soaGlobalCP	sgcp	false	none	Location of optional global configuration plan.	The absolute path to an optional global configuration plan.
-skipFileSetCapture	sf	false	none	N/A	If specified, file sets are not captured for the component during introspection.
-sudoUser	su	false	none	User name of sudo user.	Specifies a sudo user. When specified, remote capturing of file sets or introspection will substitute the user (sudo) as the sudo user before running the remote Assembly Builder. If sudoUser is specified, you cannot use the privateKeyFile. That is, sudoUser can only be used when you provide a password.
-wlsHome	wh	true	none	WLS Home	Full path to the Oracle WebLogic Server home (usually, <middleware home>/wlserver_10.3).

A.1.97.4 Examples

Here are some command examples.

A.1.97.4.1 Basic Introspection

This is a basic introspection of a Reports appliance.

```
abctl introspectReports <Reports options>
```

A.1.97.4.2 Introspection with a specific capture name

An introspection of component "Reports" saved with an appliance/assembly name of "myIntrospection":

```
% abctl introspectReports -name myIntrospection <Reports options>
```

A.1.98 introspectSIDB

Details for this command follow.

A.1.98.1 Synopsis

```
introspectSIDB -dbHome Path -oracleSid String [-name String] [-force] [-noing]
[-remoteHost String] [-remoteUser String] [-remoteWorkingDir Path] -shutdownDBOK
String [-remoteCleanup] [-dataFileDir Path] [-flashRecoveryDir Path] [-sudoUser]
[-description]
```

A.1.98.2 Description

Examines single-instance Oracle database (releases 10.2, 11.1, 11.2) configuration and captures metadata.

A.1.98.3 Options

Table A–95 shows the command options for introspectSIDB.

Table A–95 introspectSIDB options

Name	Alias	Req'd	Default Values	Possible Values	Description
-dataFileDir	dfd	false	DB 10.2 release: <Parent directory of $ORACLE_HOME>/oradata DB 11.1 and 11.2 release: $ORACLE_BASE/oradata	N/A	The full path of the database files. This parameter is required if your database file directory is different from the default.
-dbHome	dh	true	none	N/A	The ORACLE_HOME of the Oracle RDBMS to be introspected.
-description	d	false	none	N/A	Description of the appliance or assembly being introspected.
-flashRecoveryDir	frd	false	DB 10.2 release: <Parent directory of $ORACLE_HOME>/flash_recovery_area DB 11.1 release: $ORACLE_BASE/flash_recovery_area DB 11.2 release: $ORACLE_BASE/recovery_area	N/A	The full path of the database flash recovery files. This parameter is required if your recovery area is different from the default. If you do not have a recovery area, you can ignore this parameter.
-force	f	false	none	N/A	Overwrite any introspection in the catalog that exists with the same name.
-name	n	false	Derived directory name prefixed by component type name.	Any name not already used within the catalog.	Specifies a name by which the introspection output is stored
-oracleSid	os	true	none	N/A	The SID of the Oracle RDBMS to be introspected.
-privateKeyFile	pkf	false	none	N/A	Private SSH key file on the local machine.
-remoteCleanup	rc	false	none	N/A	Remote clean up flag. When set, the remote working directory will be deleted after the operation. Otherwise the directory will not be modified. If set, remoteUser and remoteHost must be specified as well.
-remoteHost	rh	false	none	N/A	Host name or IP address and optional SSH port of the remote machine. If set, remoteUser must be specified as well.
-remoteUser	ru	false	none	N/A	Name of the ssh user to use for accessing the remote machine. If set, remoteHost must be specified as well.
-remoteWorkingDir	rwd	false	/tmp/abRemote_<remote user name>	N/A	Path on the remote machine to work out of. If set, remoteUser and remoteHost must be specified as well.

Table A–95 (Cont.) introspectSIDB options

Name	Alias	Req'd	Default Values	Possible Values	Description
-shutdownDBOK	sdbok	true	none	N/A	This flag needs to be passed to approve the database reboot.
-skipFileSetCapture	sf	false	none	N/A	If specified, file sets are not captured for the component during introspection.
-sudoUser	su	false	none	User name of sudo user.	Specifies a sudo user. When specified, remote capture of file sets or introspection will substitute the user (sudo) as the sudo user before running the remote Assembly Builder. If sudoUser is specified, you cannot use the privateKeyFile. That is, sudoUser can only be used when you provide a password.

A.1.98.4 Examples

Here are some command examples.

A.1.98.4.1 Basic Introspection

This is a basic introspection of a single-instance DB appliance.

```
abctl introspectSIDB <DB options>
```

A.1.98.4.2 Introspection into a specific catalog with a specific capture name

```
% abctl introspectSIDB -name myIntrospection <DB options>
```

A.1.99 introspectSOA

Details for this command follow.

A.1.99.1 Synopsis

```
introspectSOA -wlsHome Path -domainRoot Path -adminUser String [-soaGlobalCP Path]
[-name String] [-force] [-skipFileSetCapture] [-remoteHost String] [-remoteUser
String] [-sudoUser String] [-remoteWorkingDir Path] [-remoteCleanup]
[-privateKeyFile Path] [-description]
```

A.1.99.2 Description

This command is an alias for `introspectWLS`. Examines the configuration of an installed WebLogic domain to determine what file set needs to be captured and what configuration needs to be changed at deployment.

All collected data is stored in the catalog upon successful completion.

A.1.99.3 Options

Table A–96 shows the command options for `introspectSOA`.

Table A–96 introspectSOA options

Name	Alias	Req'd	Default Values	Possible Values	Description
-adminUser	au	true	none	N/A	Administrative name for the WebLogic domain.
-description	d	false	none	N/A	Description of the appliance or assembly being introspected.
-domainRoot	dr	true	none	N/A	Full path to the WebLogic domain root.
-force	f	false	none	N/A	Overwrite any introspection in the catalog that exists with the same name.
-name	n	false	Derived directory name prefixed by component type name.	Any name not already used within the catalog.	Specifies a name by which the introspection output is stored.
-privateKeyFile	pkf	false	none	N/A	Private SSH key file on the local machine.
-remoteCleanup	rc	false	false	N/A	Remote clean up flag. When set, the remote working directory will be deleted after the operation. Otherwise the directory will not be modified. If set, remoteUser and remoteHost must be specified as well.
-remoteHost	rh	false	none	N/A	Host name or IP address and optional SSH port of the remote machine. If set, remoteUser must be specified as well.
-remoteUser	ru	false	none	N/A	Name of the SSH user to use for accessing the remote machine. If set, remoteHost must be specified as well.
-remoteWorkingDir	rwd	false	/tmp/abRemote_<remote user name>	N/A	Path on the remote machine to work out of. If set, remoteUser and remoteHost must be specified as well.
-soaGlobalCP	sgcp	false	none	Location of optional global configuration plan.	The absolute path to an optional global configuration plan.
-skipFileSetCapture	sf	false	none	N/A	If specified, file sets are not captured for the component during introspection.
-sudoUser	su	false	none	User name of sudo user.	Specifies a sudo user. When specified, remote capturing of file sets or introspection will substitute the user (sudo) as the sudo user before running the remote Assembly Builder. If sudoUser is specified, you cannot use the privateKeyFile. That is, sudoUser can only be used when you provide a password.
-wlsHome	wh	true	none	WebLogic Home	Full path to the WebLogic home directory (usually, <middleware home>/wlserver).

A.1.99.4 Examples

Here are some command examples.

A.1.99.4.1 Basic Introspection

This is a basic introspection of a single-instance DB appliance.

```
abctl introspectSOA <SOA options>
```

A.1.99.4.2 Introspection with a specific capture name

```
% abctl introspectSOA -name myIntrospection <SOA options>
```

A.1.100 introspectTuxedo

Details for this command follow.

A.1.100.1 Synopsis

```
introspectTuxedo -TUXDIR Path -TUXCONFIG Path [-environmentScript Path]
[-oracleClientDir Path] [-tnsNamesLocation Path] [-artCICSAppHome Path]
[-artBatchSecurityProfile Path] [-name String] [-force] [-skipFileSetCapture]
[-remoteHost String] [-remoteUser String] [-sudoUser String] [-remoteWorkingDir
Path] [-remoteCleanup] [-privateKeyFile Path] [-description]
```

A.1.100.2 Description

Examines a single or multiple-machine Oracle Tuxedo domain, and the Oracle Home Directory that it resides on (including add-ons).

A.1.100.3 Options

Table A–97 shows the command options for `introspectTuxedo`.

Table A–97 introspectTuxedo options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-artCICSBatchHome`	acicsah	false	none	N/A	The absolute path to the ART CICS Application Home.
`-artBatchSecurityProfile`	absp	false	none	N/A	The absolute path to the security_profile for ART Batch.
`-description`	d	false	none	N/A	Description of the appliance or assembly being introspected.
`-environmentScript`	es	false	none	The absolute path to the environment script of the application to introspect	The absolute path to the script that sets the environment of the Tuxedo application to introspect.
`-force`	f	false	none	N/A	Overwrite any introspection in the catalog that exists with the same name.
`-name`	n	false	Derived directory name prefixed by component type name.	Any name not already used within the catalog.	Specifies a name by which the introspection output is stored.
`-oracleClientDir`	ocd	false	none	The absolute path to the Oracle Database Client software.	The absolute path to the location where the Oracle Database Client software is installed.
`-privateKeyFile`	pkf	false	none	N/A	Private SSH key file on the local machine.
`-remoteCleanup`	rc	false	false	N/A	Remote clean up flag. When set, the remote working directory will be deleted after the operation. Otherwise the directory will not be modified. If set, `remoteUser` and `remoteHost` must be specified as well.
`-remoteHost`	rh	false	none	N/A	Host name or IP address and optional SSH port of the remote machine. If set, `remoteUser` must be specified as well.
`-remoteUser`	ru	false	none	N/A	Name of the SSH user to use for accessing the remote machine. If set, `remoteHost` must be specified as well.

Table A–97 (Cont.) introspectTuxedo options

Name	Alias	Req'd	Default Values	Possible Values	Description
-remoteWorkingDir	rwd	false	/tmp/abRemote_<remote user name>	N/A	Path on the remote machine to work out of. If set, remoteUser and remoteHost must be specified as well.
-soaGlobalCP	sgcp	false	none	Location of optional global configuration plan.	The absolute path to an optional global configuration plan.
-skipFileSetCapture	sf	false	none	N/A	If specified, file sets are not captured for the component during introspection.
-sudoUser	su	false	none	User name of sudo user.	Specifies a sudo user. When specified, remote capturing of file sets or introspection will substitute the user (sudo) as the sudo user before running the remote Assembly Builder. If sudoUser is specified, you cannot use the privateKeyFile. That is, sudoUser can only be used when you provide a password.
-tnsNamesLocation	tnl	false	none	The absolute path to the TNSNAMES.ora file.	The absolute path to the location of the TNSNAMES.ora file.
-TUXDIR	tuxdir	true	none	N/A	The absolute path to the TUXDIR to introspect.
-TUXCONFIG	tuxconfig	true	none	N/A	The absolute path to the TUXCONFIG file of the application to introspect.

A.1.100.4 Examples

Here are some command examples.

A.1.100.4.1 Basic Introspection

This is a basic introspection of a single-instance DB appliance.

```
abctl introspectTuxedo <Tuxedo options>
```

A.1.100.4.2 Introspection with a specific capture name

```
% abctl introspectTuxedo -name myIntrospection <Tuxedo options>
```

A.1.101 introspectWLS

Details for this command follow.

A.1.101.1 Synopsis

```
introspectWLS -wlsHome Path -domainRoot Path -adminUser String [-soaGlobalCP Path]
[-name String] [-force] [-skipFileSetCapture] [-remoteHost String] [-remoteUser
String] [-sudoUser String] [-remoteWorkingDir Path] [-remoteCleanup]
[-privateKeyFile Path] [-description]
```

A.1.101.2 Description

Examines the configuration of an installed WebLogic Server component to determine what file sets must be captured and what configuration must be changed at deployment. All collected data is stored in the catalog upon successful completion.

A.1.101.3 Extensions

The `CoherenceWeb`, `SOACoherence`, and SOA extensions are available as alias commands.

A.1.101.4 CoherenceWeb Extension Description

Inspects and captures Coherence cluster and cache server configuration that is defined within a WebLogic domain configuration. This extension supports the out-of-process deployment topology in which cache servers run in their own processes.

A.1.101.5 SOACoherence Extension Description

Inspects and captures Coherence configuration specified within SOA Managed Server start arguments to enable SOA cluster high availability.

A.1.101.6 SOA Extension Description

Oracle SOA platform plug-in. Examines the configuration of an installed WebLogic domain to determine what file set needs to be captured and what configuration needs to be changed at deployment.

A.1.101.7 Options

Table A–98 shows the command options for `introspectWLS`.

Table A–98 introspectWLS options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-adminUser`	au	true	none	N/A	Administrative name for the WebLogic domain.
`-description`	d	false	none	N/A	Description of the appliance or assembly being introspected.
`-domainRoot`	dr	true	none	N/A	Full path to the WebLogic domain root directory.
`-force`	f	false	none	N/A	Overwrite any introspection in the catalog that exists with the same name.
`-name`	n	false	Derived directory name prefixed by component type name.	Any name not already used within the catalog.	Specifies a name by which the introspection output is stored.
`-privateKeyFile`	pkf	false	none	N/A	Private SSH key file on the local machine.
`-remoteCleanup`	rc	false	false	N/A	Remote clean up flag. When set, the remote working directory will be deleted after the operation. Otherwise the directory will not be modified. If set, `remoteUser` and `remoteHost` must be specified as well.
`-remoteHost`	rh	false	none	N/A	Host name or IP address and optional SSH port of the remote machine. If set, `remoteUser` must be specified as well.
`-remoteUser`	ru	false	none	N/A	Name of the SSH user to use for accessing the remote machine. If set, `remoteHost` must be specified as well.
`-remoteWorkingDir`	rwd	false	`/tmp/abRemote_<remote user name>`	N/A	Path on the remote machine to work out of. If set, `remoteUser` and `remoteHost` must be specified as well.
`-skipFileSetCapture`	sf	false	none	N/A	If specified, file sets are not captured for the component during introspection.

Table A–98 (Cont.) introspectWLS options

Name	Alias	Req'd	Default Values	Possible Values	Description
-soaGlobalCP	sgcp	false	none	Location of optional global configuratio n plan.	The absolute path to an optional global configuration plan.
-sudoUser	su	false	none	User name of sudo user.	Specifies a sudo user. When specified, remote capturing of file sets or introspection will substitute the user (sudo) as the sudo user before running the remote Assembly Builder. If sudoUser is specified, you cannot use the privateKeyFile. That is, sudoUser can only be used when you provide a password.
-wlsHome	wh	true	none	WebLogic Home	Full path to the WebLogic home directory (usually, <middleware home>/wlserver).

A.1.101.8 Examples

Here are some command examples.

A.1.101.8.1 Successful Introspection: local execution with use of all options

```
% abctl introspectWLS -name myWlsCapture
-wlsHome /ora/mw/wlserver -domainRoot /ora/mw/user_projects/domains/MyDomain
-adminUser weblogic
Launching introspection of appliance 'WLS' ...
  Step 1 of 3: WLS dehydration starting. Due to domain template creation this may
take some time
    Step 1 of 15: WlsAssemblyBuilder has started creating the AssemblyBuilder
      Step 1 of 2: Capturing Node Manager configuration.
      Step 2 of 2: Node Manager capture complete.
    Step 12 of 15: Processor: 10 completed
    Step 15 of 15: WlsAssemblyBuilder has completed the AssemblyBuilder
  Step 2 of 3: WLS Assembly is completed
  Step 3 of 3: WLS dehydration completed
Task is done: DehydrateJob completed
Introspection complete
Storing result in catalog: ...
Introspection stored as 'myWlsCapture' in the catalog
%
```

A.1.101.8.2 Successful Introspection: local execution with all defaults and short names

```
% abctl introspectWLS -adminUser weblogic -wh /ora/mw/wlserver
-dr /ora/mw/user_projects/domains/MyDomain
Launching introspection of appliance 'WLS' ...
  Step 1 of 3: WLS dehydration starting. Due to domain template creation this may
take some time
    Step 1 of 15: WlsAssemblyBuilder has started creating the AssemblyBuilder
      Step 1 of 2: Capturing Node Manager configuration.
      Step 2 of 2: Node Manager capture complete.
    Step 12 of 15: Processor: 10 completed
    Step 15 of 15: WlsAssemblyBuilder has completed the AssemblyBuilder
  Step 2 of 3: WLS Assembly is completed
  Step 3 of 3: WLS dehydration completed
Task is done: DehydrateJob completed
Introspection complete
Storing result in catalog: '/ora/ab/catalog' ...
Introspection stored as 'WLS-1256089687424' in the catalog
```

```
%
```

A.1.101.8.3 Missing -wlsHome Parameter

```
% abctl introspectWLS -domainRoot
/ora/mw/user_projects/domains/MyDomain
Error: missing required parameter 'wlsHome'

Command usage:

introspectWLS [-name string]
   [-remoteHost string] [-remotePort numeric] [-remoteUser string]
   [-remoteWorkingDir path] -wlsHome path -domainRoot path

Try 'abctl help -command introspectWLS' for detailed help of the command.

%
```

A.1.101.8.4 Bad -domainRoot path

```
$ abctl introspectWLS -adminUser weblogic -wlsHome
/scratch/aime1/Oracle/Middleware/wlserver/ -domainRoot /tmp/foobar -name test
Enter 'Admin Password':
Launching introspection of appliance 'WLS' ...
 Step 1 of 3: Started WLS dehydration (expect delays during domain template
creation)..
Task is done: Dehydration failed with error: The domainRoot specified does not
exist..
Error: OAB-7105: Introspection failed.
Caused by: OAB-50005: The domainRoot specified does not exist.
```

A.1.102 redeployAssemblyInstance

Details for this command follow.

A.1.102.1 Synopsis

```
$ abctl redeployAssemblyInstance -assemblyInstanceId String -connectionName String
[-waitForComplete] [-pollTime String]
```

A.1.102.2 Description

Redeploys an assembly instance, and is equivalent to performing an `undeployAssemblyInstance` followed by a `deployAssemblyInstance`.

A.1.102.3 Options

Table A–99 shows the command options for `redeployAssemblyInstance`.

Table A–99 redeployAssemblyInstance options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-assemblyInstanceId`	a	true	none	A string representing the `assemblyInstanceId`.	The identifier of an assembly instance to redeploy.

Table A–99 (Cont.) redeployAssemblyInstance options

Name	Alias	Req'd	Default Values	Possible Values	Description
-connection Name	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
-pollTime	pt	false	5	A string representing the number of seconds.	Specifies the amount of time to wait for an asynchronous operation to complete successfully.
-waitForComplete	w	false	no	N/A	Specifies whether to wait for an asynchronous operation to complete successfully.

A.1.102.4 Examples

Here are some command examples.

A.1.102.4.1 Redeploy Deployment

```
$ abctl redeployAssemblyInstance -assemblyInstanceId MyId
```

A.1.103 registerAssemblyArchive

Details for this command follow.

A.1.103.1 Synopsis

```
$ abctl registerAssemblyArchive -name String [-version String] [-target String]
-connectionName String [-waitForComplete] [-pollTime String]
```

A.1.103.2 Description

Registers an assembly archive in the Deployer.

A.1.103.3 Options

Table A–100 shows the command options for `registerAssemblyArchive`.

Table A–100 registerAssemblyArchive options

Name	Alias	Req'd	Default Values	Possible Values	Description
-connection Name	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
-name	n	true	none	A string representing the name of the assembly archive.	The name of the assembly archive.
-pollTime	pt	false	5	A string representing the number of seconds.	Specifies the amount of time to wait for an asynchronous operation to complete successfully.
-target	t	false	none	A string representing the name of the target.	The name of the target.
-version	v	false	none	A string representing the version of the assembly archive.	The version of the assembly archive.
-waitForComplete	w	false	no	N/A	Specifies whether to wait for an asynchronous operation to complete successfully.

A.1.103.4 Examples

Here are some command examples.

A.1.103.4.1 Register Assembly Archive

```
$ abctl registerAssemblyArchive -connectionName MyDeployerConnection -name
TheAssembly -version 1
```

A.1.104 removeAssemblyUsers

Details for this command follow.

A.1.104.1 Synopsis

```
$ abctl removeAssemblyUsers -assembly String -user String... -connectionName
String
```

A.1.104.2 Description

Removes one or more users from an assembly.

A.1.104.3 Options

Table A–103 shows the command options for `removeAssemblyUsers`.

Table A–101 getDefaultTarget options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-assembly`	a	false	none	A string representing the name of the assembly archive.	Specifies the assembly to remove users from.
`-connection Name`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
`-user`	u	true	none	A string representing the usernames of the users to remove from an assembly archive.	The usernames of the users to remove from an assembly archive.

A.1.104.4 Examples

Here are some command examples.

A.1.104.4.1 Remove Assembly Users

```
$ abctl removeAssemblyUsers -assembly MyAssembly -user User1 User2
```

A.1.105 removePlugin

Details for this command follow.

A.1.105.1 Synopsis

```
$ abctl removePlugin [-pluginName] String
```

A.1.105.2 Description

Recursively removes the specified plug-in or extension and any child extensions.

A.1.105.3 Options

Table A–102 shows the command options for `removePlugin`.

Table A–102 removePlugins options

Name	Alias	Req'd	Default Values	Possible Values	Description
-pluginName	pn	true	none	N/A	A single plug-in name.

A.1.105.4 Examples

Here are some command examples.

A.1.105.4.1 Remove Plug-in

```
$ abctl removePlugin WLS
```

A.1.106 removeTargetUsers

Details for this command follow.

A.1.106.1 Synopsis

```
$ abctl removeTargetUsers -user String... -target String -connectionName String
```

A.1.106.2 Description

This command removes a user from the target.

A.1.106.3 Options

Table A–103 shows the command options for `removeTargetUsers`.

Table A–103 getDefaultTarget options

Name	Alias	Req'd	Default Values	Possible Values	Description
-connection Name	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
-target	t	true	none	A string representing the target to add the user to.	The target to add the user to.
-user	u	true	none	A string representing the usernames of the users to remove from the target	The usernames of the users to remove from the target.

A.1.106.4 Examples

Here are some command examples.

A.1.106.4.1 Remove Target Users

```
$ abctl removeTargetUsers -user Username -target Targetname
```

A.1.107 renameExternalResource

Details for this command follow.

A.1.107.1 Synopsis

```
$ abctl renameExternalResource -currentName String -newName String -assembly
String
```

A.1.107.2 Description

Renames the specified external resource. Only external resources can be renamed.

A.1.107.3 Options

Table A–104 shows the command options for `renameExternalResource`.

Table A–104 renameExternalResource options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-currentName`	cn	true	none	N/A	The name of the external resource to be renamed.
`-newName`	nn	true	none	N/A	The new name for the external resource.
`-assembly`	as	true	none	N/A	The assembly containing the external resource to be renamed.

A.1.107.4 Examples

Here are some command examples.

A.1.107.4.1 Rename External Resource

```
% abctl renameExternalResource -assembly mySite -currentName jdbc0 -newName
myDatabase
```

A.1.108 restartAssemblyInstance

Details for this command follow.

A.1.108.1 Synopsis

```
$ abctl restartAssemblyInstance -assemblyInstanceId String -connectionName String
[-waitForComplete] [-pollTime String]
```

A.1.108.2 Description

This command restarts an assembly instance, and is equivalent to performing a `stopAssemblyInstance` followed by a `startAssemblyInstance`.

A.1.108.3 Options

Table A–105 shows the command options for `restartAssemblyInstance`.

Table A–105 restartAssemblyInstance options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-assemblyInstanceId`	a	true	none	A string representing the `assemblyInstanceId`.	The identifier of an assembly instance to be restarted.
`-connectionName`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
`-pollTime`	t	false	5	A string representing the number of seconds.	Specifies the amount of time to wait for an asynchronous operation to complete successfully.
`-waitForComplete`	w	false	no	N/A	Specifies whether to wait for an asynchronous operation to complete successfully.

A.1.108.4 Examples

Here are some command examples.

A.1.108.4.1 Restart Deployment

```
$ abctl restartAssemblyInstance -assemblyInstanceId MyId
```

A.1.109 resumeAssemblyInstance

Details for this command follow.

A.1.109.1 Synopsis

```
$ abctl resumeAssemblyInstance -assemblyInstanceId String -connectionName String
[-waitForComplete] [-pollTime Numeric]
```

A.1.109.2 Description

This command resumes a suspended assembly instance.

A.1.109.3 Options

Table A–106 shows the command options for `resumeAssemblyInstance`.

Table A–106 resumeAssemblyInstance options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-assemblyInstanceId`	a	true	none	A string representing the `assemblyInstanceId`.	The identifier of an assembly instance.
`-connectionName`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
`-waitForComplete`	w	false	no	N/A	Specifies whether to wait for an asynchronous operation to complete successfully.
`-pollTime`	pt	false	5	A string representing the number of seconds.	Specifies the amount of time to wait for an asynchronous operation to complete successfully.

A.1.109.4 Examples

Here are some command examples.

A.1.109.4.1 Resume Assembly Instance

```
$ abctl resumeAssemblyInstance -assemblyInstanceId MyId
```

A.1.110 scale

Details for this command follow.

A.1.110.1 Synopsis

```
$ abctl scale -scalingGroupId String -target String -connectionName String
[-waitForComplete] [-pollTime String]
```

A.1.110.2 Description

Scales a scaling group to a new size.

A.1.110.3 Options

Table A–107 shows the command options for `scale`.

Table A–107 scale options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-connectionName`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
`-pollTime`	pt	false	5	A string representing the number of seconds.	Specifies the amount of time to wait for an asynchronous operation to complete successfully.
`-scalingGroupId`	s	true	none	A string representing the `scalingGroupId`.	The `scalingGroupId` of a scalingGroup.
`-target`	t	true	none	A string representing the new target.	The new value to scale to.
`-waitForComplete`	w	false	no	N/A	Specifies whether to wait for an asynchronous operation to complete successfully.

A.1.110.4 Examples

Here are some command examples.

A.1.110.4.1 Scale a Scaling Group

```
$ abctl scale -scalingGroupId FOO -target 4
```

A.1.111 scaleDown

Details for this command follow.

A.1.111.1 Synopsis

```
$ abctl scaleDown -applianceInstanceIds String -target String -connectionName
String [-waitForComplete]
```

A.1.111.2 Description

Scales the specified appliance instance IDs to a new size.

A.1.111.3 Options

Table A–108 shows the command options for `scaleDown`.

Table A–108 scaleDown options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-connectionName`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
`-target`	t	true	none	A string representing the new target.	The new value to scale to.
`-waitForComplete`	w	false	no	N/A	Specifies whether to wait for an asynchronous operation to complete successfully.

A.1.111.4 Examples

Here are some command examples.

A.1.111.4.1 Scale Down

```
$ abctl scaleDown -applianceInstanceIds
BNbV9P5Td-deployAssembly:deployAssembly/prod0/1 -connectionName adminConn
-waitForComplete
```

A.1.112 setCatalogProperty

Details for this command follow.

A.1.112.1 Synopsis

```
$ abctl setCatalogProperty -containerAddress String -name String -value String
```

A.1.112.2 Description

This command allows a property value to be updated within the catalog metadata. Not all properties in the catalog are are settable.

A.1.112.3 Options

Table A–109 shows the command options for `setCatalogProperty`.

Table A–109 setCatalogProperty options

Name	Alias	Req'd	Default Values	Possible Values	Description
-containerAddress	c	true	none	A variable number of arguments.	Takes a variable number of arguments to specify a single container (such as an assembly, appliance, vnet, or physical interface) within the catalog's metadata hierarchy. It comprises a series of 'steps' one would take for navigating down through this hierarchy, starting from the root (catalog) level. The argument follows a Unix-like syntax although it does not refer to an actual directory path. It always begin with a slash ("/") character. For instance, to specify the Resource Requirements of the AdminServer appliance of a Weblogic server named wls1 of an assembly named "mySite", the container address is: /assemblies/mySite/assemblies/wls1/appliances/AdminServer/resource-requirements
-name	n	true	none	A string representing the name of a single property.	Specifies a single property in the output. This is useful primarily to enable a script to extract (scrape) a specific value from stdout. If the name parameter is omitted, all properties and child containers of the specified container are displayed.
-value	v	true	none	A string representing the value to be assigned.	Specifies a string representation of the value to be assigned to the property being updated. Assigning an empty string ("") clears the current value, regardless of the property type. For security reasons, passwords cannot be entered directly in the command itself. To assign a new value to a password property, specify a non-blank value (such as "xxx") to activate password prompting. Assigning an empty string ("") clears the password without prompting.

A.1.112.4 Examples

Here are some command examples.

A.1.112.4.1 Set Catalog Property

```
$ abctl setCatalogProperty -ca /appliances/ohs1/resource-requirements -name
memory-mb -value 2048
Successfully set property memory-mb
```

A.1.112.4.2 Clearing a Property Value

```
$ abctl setCatalogProperty -ca
/assemblies/wls1/appliances/AdminServer/inputs/Default -name port -value ""
```

```
Successfully set property port
```

A.1.112.4.3 Setting a Password Property

```
$ abctl setCatalogProperty -ca /assemblies/wls1/user-properties -name
admin-password -value xxx
 Enter password for property admin-password :
 Enter same passord to confirm :
 Successfully set property admin-password.
```

A.1.113 setDefaultTarget

Details for this command follow.

A.1.113.1 Synopsis

```
$ abctl setDefaultTarget -name String -connectionName String
```

A.1.113.2 Description

This command sets a target as the default.

A.1.113.3 Options

Table A–110 shows the command options for `setDefaultTarget`.

Table A–110 setDefaultTarget options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-connection Name`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
`-name`	n	true	none	A string representing the name of the target.	`The name of the target.`

A.1.113.4 Examples

Here are some command examples.

A.1.113.4.1 Set Default Target

```
$ abctl setDefaultTarget -name MyTarget
```

A.1.114 startAssemblyInstance

Details for this command follow.

A.1.114.1 Synopsis

```
$ abctl startAssemblyInstance -assemblyInstanceId String -connectionName String
[-waitForComplete] [-pollTime String]
```

A.1.114.2 Description

Starts an assembly instance.

A.1.114.3 Options

Table A–111 shows the command options for `startAssemblyInstance`.

Table A–111 startAssemblyInstance options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-assemblyInstanceId`	a	true	none	A string representing the `assemblyInstanceId`.	The identifier of an assembly instance to be started.
`-connectionName`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
`-pollTime`	pt	false	5	A string representing the number of seconds.	Specifies the amount of time to wait for an asynchronous operation to complete successfully.
`-waitForComplete`	w	false	no	N/A	Specifies whether to wait for an asynchronous operation to complete successfully.

A.1.114.4 Examples

Here are some command examples.

A.1.114.4.1 Start Assembly Instance

```
$ abctl startAssemblyInstance
```

A.1.115 stopAssemblyInstance

Details for this command follow.

A.1.115.1 Synopsis

```
$ abctl stopAssemblyInstance -assemblyInstanceId String [-force] -connectionName
String [-waitForComplete] [-pollTime String]
```

A.1.115.2 Description

This command stops a deployment for an assembly instance.

A.1.115.3 Options

Table A–112 shows the command options for `stopAssemblyInstance`.

Table A–112 stopAssemblyInstance options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-assemblyInstanceId`	a	true	none	A string representing the assemblyInstanceId.	The identifier of an assembly instance to be stopped.
`-connectionName`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
`-force`	f	false	none	True/false.	Flag to indicate if local cleanup should be done even if the resource manager is not available.
`-pollTime`	pt	false	5	A string representing the number of seconds.	Specifies the amount of time to wait for an asynchronous operation to complete successfully.
`-waitForComplete`	w	false	no	N/A	Specifies whether to wait for an asynchronous operation to complete successfully.

A.1.115.4 Examples

Here are some command examples.

A.1.115.4.1 Stop Assembly Instance

```
$ abctl stopAssemblyInstance
```

A.1.116 suspendAssemblyInstance

Details for this command follow.

A.1.116.1 Synopsis

```
$ abctl suspendAssemblyInstance -assemblyInstanceId String -connectionName String
[-waitForComplete] [-pollTime Numeric]
```

A.1.116.2 Description

This command suspends an assembly instance.

A.1.116.3 Options

Table A–113 shows the command options for `resumeAssemblyInstance`.

Table A–113 suspendAssemblyInstance options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-assemblyInstanceId`	a	true	none	A string representing the `assemblyInstanceId`.	The identifier of an assembly instance.
`-connectionName`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
`-waitForComplete`	w	false	no	N/A	Specifies whether to wait for an asynchronous operation to complete successfully.
`-pollTime`	pt	false	5	A string representing the number of seconds.	Specifies the amount of time to wait for an asynchronous operation to complete successfully.

A.1.116.4 Examples

Here are some command examples.

A.1.116.4.1 Suspend Assembly Instance

```
$ abctl suspendAssemblyInstance -assemblyInstanceId MyId
```

A.1.117 unbindAssemblyFileSetDefinition

Details for this command follow.

A.1.117.1 Synopsis

```
$ abctl unbindAssemblyFileSetDefinition -name String -from String
```

A.1.117.2 Description

Unbinds an assembly file set definition from an appliance. Unbinding it from one appliance in an atomic assembly unbinds it from all appliances in that atomic assembly.

A.1.117.3 Options

Table A–114 shows the command options for `unbindAssemblyFileSetDefinition`.

Table A–114 unbindAssemblyFileSetDefinition options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-name`	n	true	none	N/A	Name of file set definition on top level, non-atomic assembly.
`-from`	fr	true	none	N/A	Path to nested appliance.

A.1.117.4 Examples

Here are some command examples.

A.1.117.4.1 Bind Assembly File Set Definition

```
% abctl unbindAssemblyFileSetDefinition -name middleware -from mySite/ohs1
```

A.1.118 undeployAssemblyInstance

Details for this command follow.

A.1.118.1 Synopsis

```
$ abctl undeployAssemblyInstance -assemblyInstanceId String -connectionName String
[-waitForComplete] [-pollTime String]
```

A.1.118.2 Description

Undeploys an assembly instance.

A.1.118.3 Options

Table A–115 shows the command options for `undeployAssemblyInstance`.

Table A–115 undeployAssemblyInstance options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-assemblyInstanceId`	d	true	none	A string representing the `assemblyInstanceId`.	The identifier of an assembly instance to undeploy.
`-connectionName`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
`-pollTime`	pt	false	5	A string representing the number of seconds.	Specifies the amount of time to wait for an asynchronous operation to complete successfully.
`-waitForComplete`	w	false	no	N/A	Specifies whether to wait for an asynchronous operation to complete successfully.

A.1.118.4 Examples

Here are some command examples.

A.1.118.4.1 Undeploy Assembly Instance

```
$ abctl undeployAssemblyInstance -assemblyInstanceId MyId
```

A.1.119 unregisterAssemblyArchive

Details for this command follow.

A.1.119.1 Synopsis

```
$ abctl unregisterAssemblyArchive -name String [-version String] [-target String]
-connectionName String [-waitForComplete] [-pollTime String]
```

A.1.119.2 Description

This command unregisters an assembly from the Deployer.

A.1.119.3 Options

Table A–116 shows the command options for `unregisterAssemblyArchive`.

Table A–116 unregisterAssemblyArchive options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-connection Name`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
`-name`	n	true	none	A string representing the name of the assembly archive.	The name of the assembly archive.
`-pollTime`	pt	false	5	A string representing the number of seconds.	Specifies the amount of time to wait for an asynchronous operation to complete successfully.
`-target`	t	false	none	A string representing the name of the target.	The name of the target.
`-version`	v	false	none	A string representing the version of the assembly archive.	The version of the assembly archive.
`-waitForCom plete`	w	false	no	N/A	Specifies whether to wait for an asynchronous operation to complete successfully.

A.1.119.4 Examples

Here are some command examples.

A.1.119.4.1 Unregister Assembly Archive

```
$ abctl unregisterAssemblyArchive -name TheAssembly -version 1
```

A.1.120 updateAssemblyArchive

Details for this command follow.

A.1.120.1 Synopsis

```
$ abctl updateAssemblyArchive -name String -version String -description String
-connectionName String
```

A.1.120.2 Description

Updates the description (attributes) of an assembly archive in the Deployer.

A.1.120.3 Options

Table A–117 shows the command options for `updateAssemblyArchive`.

Table A–117 updateAssemblyArchive options

Name	Alias	Req'd	Default Values	Possible Values	Description
-connectionName	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
-description	d	true	none	A string representing the description of the assembly archive.	The description of the assembly archive.
-name	n	true	none	A string representing the name of the assembly archive.	The name of the assembly archive.
-version	v	true	none	A string representing the version of the assembly archive.	The version of the assembly archive.

A.1.120.4 Examples

Here are some command examples.

A.1.120.4.1 Update Assembly Archive

```
$ abctl updateAssemblyArchive -name MyAssembly -version 1 -description
NewDescription
```

A.1.121 updateDeployerConfig

Details for this command follow.

A.1.121.1 Synopsis

```
$ abctl updateDeployerConfig -properties String... [-domainRoot Path]
```

A.1.121.2 Description

Updates the configuration of the deployer. Configuration properties supported are: `ovab.webserver.url` and `ovab.directory`.

A.1.121.3 Options

Table A–118 shows the command options for `updateDeployerConfig`.

Table A–118 updateDeployerConfig options

Name	Alias	Req'd	Default Values	Possible Values	Description
-properties	p	true	none	A string representing the properties and values to update.	A set of key=value pairs representing the property and its new value.
-domainRoot	dr	false	none	A string representing the root directory of the Deployer domain.	The root directory of the Deployer domain.

A.1.121.4 Examples

Here are some command examples.

A.1.121.4.1 Update Deployer Configuration

```
$ abctl updateDeployerConfig -properties ovab.webserver.url=http://localhost:7001
ovab.directory=c:/files
```

A.1.122 updateTarget

Details for this command follow.

A.1.122.1 Synopsis

```
$ abctl updateTarget -name String [-properties String...] -connectionName String
```

A.1.122.2 Description

Updates one or more property values. This command is enabled for Oracle VM targets.

A.1.122.3 Options

Table A–119 shows the command options for `updateTarget`.

Table A–119 updateTarget options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-connection Name`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
`-name`	n	true	none	A string representing the name of the target.	The name of the target.
`-properties`	p	false	none	A string representing the properties and values to update.	A set of key=value pairs representing the property and its new value.

A.1.122.4 Examples

Here are some command examples.

A.1.122.4.1 Update Target

```
$ abctl updateTarget -name MyTarget -properties prop=newvalue
```

A.1.123 uploadAssemblyArchive

Details for this command follow.

A.1.123.1 Synopsis

```
$ abctl uploadAssemblyArchive -fileName Path -name String [-description String]
-connectionName String
```

A.1.123.2 Description

Uploads an assembly archive to Oracle Virtual Assembly Builder Deployer.

A.1.123.3 Options

Table A–120 shows the command options for `uploadAssemblyArchive`.

Table A–120 uploadAssemblyArchive options

Name	Alias	Req'd	Default Values	Possible Values	Description
-connection Name	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.
-fileName	fn	true	none	A string representing the file path to the assembly archive on disk.	Uploads an assembly archive to the Deployer.
-name	n	true	none	A string representing the name of the assembly archive.	The name of the assembly archive.
-descriptio n	d	false	none	A string representing the description of the assembly archive.	The description of the assembly archive.

A.1.123.4 Examples

Here are some command examples.

A.1.123.4.1 Upload Assembly Archive

```
$ abctl uploadAssemblyArchive -fileName=c:/mySite.ova -name TheAssembly -version 1
```

A.1.124 uploadAssemblyResources

Details for this command follow.

A.1.124.1 Synopsis

```
$ abctl uploadAssemblyResources -fileName Path -assemblyName String -version
String -connectionName String [-append]
```

A.1.124.2 Description

Uploads an assembly resources file to associate with a specific version of an assembly. The resource zip file is uploaded and extracted into the repository.

The `uploadAssemblyResources` command is controlled by a security policy. A resources file may or may not contain scripts. If the resource file does not contain scripts, a user on the assembly access list can run the command. If the resource file does contain scripts, only the CloudAdmin user is allowed to run the command, to prevent a malicious attack.

When including scripts in the resources files, the lifecycle names that are supported are: `pre-deploy`, `post-deploy`, `deployer-pre-app-config`, `deployer-post-app-config`, `deployer-pre-vm-start`, `deployer-post-vm-start`, `deployer-pre-vm-stop`, `deployer-post-vm-stop`, `pre-undeploy`, `post-undeploy`. You can create corresponding script folder names.

The following is a sample resource zip file containing scripts:

```
unzip ../myResources.zip
Archive:  ../myResources.zip
    creating: disks/
   inflating: disks/test1.iso
   inflating: disks/test2.iso
    creating: scripts.d/
    creating: scripts.d/pre-deploy.d/
   inflating: scripts.d/pre-deploy.d/00script.sh
```

```
inflating: scripts.d/pre-deploy.d/01script.sh
 creating: scripts.d/post-deploy.d/
inflating: scripts.d/post-deploy.d/00script.sh
inflating: scripts.d/post-deploy.d/01script.sh
 creating: scripts.d/deployer-pre-vm-stop.d/
inflating: scripts.d/deployer-pre-vm-stop.d/00script.sh
inflating: scripts.d/deployer-pre-vm-stop.d/01script.sh
 creating: scripts.d/post-undeploy.d/
inflating: scripts.d/post-undeploy.d/00script.sh
inflating: scripts.d/post-undeploy.d/01script.sh
```

A.1.124.3 Options

Table A–121 shows the command options for `uploadAssemblyResources`.

Table A–121 uploadAssemblyResources options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-fileName`	fn	true	none	A string representing the file path to the assembly resources files on disk.	Uploads an assembly resources file to the Deployer.
`-assemblyName`	n	true	none	A string representing the name of the assembly.	The name of the assembly.
`-version`	v	false	none	A string representing the assembly version.	The assembly version.
`-append`	a	false	none	A flag, that if set, appends the assembly resources file upload.	If set, the assembly resources file upload is appended.
`-connectionName`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.124.4 Examples

Here are some command examples.

A.1.124.4.1 Upload Assembly Resources File

```
$ abctl uploadAssemblyResources -assemblyName myAssembly -version 1 -fileName
resources.zip -connectionName myConnection

Upload File Size: 2,708
100% Complete
Assembly resources zip has been uploaded to associate with assembly myAssembly,
version 1.
```

A.1.125 uploadDeploymentPlan

Details for this command follow.

A.1.125.1 Synopsis

```
$ abctl uploadDeploymentPlan -fileName Path -assemblyName String -version String
-planName String -connectionName String
```

A.1.125.2 Description

Uploads a deployment plan for an existing assembly.

A.1.125.3 Options

Table A–122 shows the command options for `uploadDeploymentPlan`.

Table A–122 uploadDeploymentPlan options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-fileName`	fn	true	none	A path to the deployment plan file.	The file name of the deployment plan to upload.
`-assemblyName`	an	true	none	A string representing the name of the assembly.	The name of the assembly.
`-version`	v	true	none	A string representing the assembly version.	The assembly version.
`-planName`	pn	true	none	A string representing the name of the deployment plan.	The name of the deployment plan.
`-connectionName`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.125.4 Examples

Here are some command examples.

A.1.125.4.1 Upload Deployment Plan

```
$ abctl
```

A.1.126 uploadEMAssemblyArchive

Details for this command follow.

A.1.126.1 Synopsis

```
$ abctl uploadEMAssemblyArchive -name String -[description String]
```

A.1.126.2 Description

Uploads an assembly archive to the Enterprise Manager Software Library. The assembly can only be a top-level assembly, and the assembly archive must be created for the assembly.

A.1.126.3 Options

Table A–123 shows the command options for `uploadEMAssemblyArchive`.

Table A–123 uploadEMAssemblyArchive options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-name`	n	true	none	Top level appliance or assembly in the catalog.	Name of a top level appliance or assembly in the catalog.
`-description`	d	false	none	Textual description.	A description of the assembly and assembly archive.

A.1.126.4 Examples

Here is a command example.

A.1.126.4.1 uploadEMAssemblyArchive

```
% abctl uploadEMAssemblyArchive -name archiveName -description "my assembly
archive"
```

A.1.127 validateAssemblyArchiveResources

Details for this command follow.

A.1.127.1 Synopsis

```
$ abctl validateAssemblyArchiveResources -metaDataFilePath String -deploymentPlan
String -targetName String -connectionName String
```

A.1.127.2 Description

Validates whether an assembly will deploy successfully based on resource requirements.

A.1.127.3 Options

Table A–124 shows the command options for `validateAssemblyArchiveResources`.

Table A–124 validateAssemblyArchiveResources options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-metaDataFilePath`	m	true	none	A string representing path to an assembly metadata file.	A path to an assembly metadata file.
`-deploymentPlan`	d	true	none	A string representing path to an assembly deployment plan.	A path to an assembly deployment plan.
`-targetName`	t	true	none	A string representing a target to check resources against.	The target to check resources against.
`-connectionName`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.127.4 Examples

Here are some command examples.

A.1.127.4.1 Validate Assembly Archive Resources

```
$ abctl validateAssemblyArchiveResources -assemblyName MyAssembly -assemblyVersion
1 -metaDataFilePath c:/MyAssembly.ovf -deploymentPlan c:/MyPlan.xml -targetName
MyTarget
```

A.1.128 validateAssemblyInstanceResources

Details for this command follow.

A.1.128.1 Synopsis

```
$ abctl validateAssemblyInstanceResources -assemblyInstanceId String
-connectionName String
```

A.1.128.2 Description

Validates whether an assembly instance will deploy successfully based on resource requirements.

A.1.128.3 Options

Table A–125 shows the command options for `validateAssemblyInstanceResources`.

Table A–125 validateAssemblyArchiveResources options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-assemblyInstanceId`	a	true	none	A string representing the `assemblyInstanceId`.	The identifier of an assembly instance to be deleted.
`-connectionName`	c	true	none	A string representing the name of the connection to the Deployer.	The name of a connection to the Deployer.

A.1.128.4 Examples

Here are some command examples.

A.1.128.4.1 Validate Assembly Instance Resources

```
$ abctl validateAssemblyInstanceResources -assemblyInstanceId MyId
```

A.1.129 verifyEMConnection

Details for this command follow.

A.1.129.1 Synopsis

```
$ abctl verifyEMConnection -connectionURL String -connectionUser String
-namedHostCredential String -remoteUser String -remoteWorkingDir Path [-sshPort
Numeric] [-privateKeyFile Path] [-fullVerification]
```

A.1.129.2 Description

Verifies a connection to an Enterprise Manager Software Library. The default command validates the connection to EM. The `-fullVerification` option attempts to upload and download a test assembly archive.

You must specify the fully qualified hostname of the remote Enterprise Manager machine, for example `myhost.example.com` instead of `myhost`.

When you perform this command, you are prompted for a connection password.

A.1.129.3 Options

Table A–126 shows the command options for `verifyEMConnection`.

Table A–126 verifyEMConnection options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-connectionURL`	c	true	none	Hostname:port.	URL for connecting to EM Software Library.
`-connectionUser`	cu	true	none	Valid EM Software Library User.	User for EM Software Library.
`-namedHostCredential`	n	true	none	Valid Named Host Credential.	Named Host Credential.
`-remoteUser`	ru	true	none	Valid SSH user.	SSH user for connecting to the machine where the EM Software Library is located.

Table A–126 (Cont.) verifyEMConnection options

Name	Alias	Req'd	Default Values	Possible Values	Description
-remoteWorkingDir	rwd	true	none	/scratch,/home/mydir	Valid directory on EM Software Library machine, where assembly archives are uploaded and consumed. Oracle Corporation recommends that you do not use the /tmp directory.
-sshPort	s	false	none	Valid SSH port number.	SSH port for EM Software Library machine.
-privateKeyFile	pkf	false	none	~/.ssh/id_rsa, id_rsa	Local private SSH key file used for SSH to the remote EM Software Library machine.
-fullVerification	fv	false	none		When this flag is specified, the command attempts to upload and download a test assembly archive.

A.1.129.4 Examples

Here are some command examples.

A.1.129.4.1 createEMConnection

```
$ abctl createEmConnection -connectionURL emMachine:7791 -connectionUser admin
-namedHostCredential hostCredential -remoteUser mySshUser -remoteWorkingDir
/scratch/myovas [-sshPort 23] [-privateKeyFile ~/.ssh/id_rsa]
```

A.1.130 version

Details for this command follow.

A.1.130.1 Synopsis

```
$ abctl version
```

A.1.130.2 Description

Displays the Oracle Virtual Assembly Builder specification version.

A.1.130.3 Example

$ abctl version

A.2 Help

Details for this command follow.

A.2.1 Synopsis

```
$ abctl help [-command String] [-category] [-usage]
```

A.2.2 Description

With no parameters, the help command displays a list of help categories that you can query to obtain the list of available commands along with the brief description of each command.

When the -all option is specified, the help command lists all available commands with a short description of each command. When a command name is specified, then detailed Help about the specified command is printed. When a command name is

specified and the `-usage` parameter is specified, just the synopsis (argument usage) is printed.

When the `-category` option is specified, the `help` command lists the commands available under that category. Commands may appear under more than one category.

A.2.3 Options

Table A–127 shows the command options for `help`.

Table A–127 help options

Name	Alias	Req'd	Default Values	Possible Values	Description
`-command`	c	false	none	Name of a command.	Name of one of the commands listed when help is invoked without arguments. If specified, then detailed help information about the specified command is provided.
`-usage`	u	false	none		Not valid when the `-command` parameter is not specified. When this flag is specified only synopsis details are provided about the corresponding specified command.
`-category`	c	false	none		When this flag is specified, the `help` command lists the commands available under the specified category.
`-quiet`	q	false	none	N/A	By default, the command shows detailed progress/success messages. If `-quiet` is set, the command turns off verbose mode and shows only one or two progress/success messages.

A.2.4 Examples

Here are some command examples.

A.2.4.1 No Arguments

```
$ abctl help

Usage: abctl command [options]

Command                Description
-------------------------------------------------------------------------------
abctl help all             List all commands.
abctl help catalog         Manage appliances and assemblies.
abctl help deployer        Setup and manage OVAB Deployer.
abctl help deployments     Deploy and manage assembly deployments.
abctl help EM              Manage assemblies in EM Software Library.
abctl help general         General help commands.
abctl help introspection   Capture product installations.
abctl help targets         Configure and manage deployment targets.

Try "abctl help -command cmd_name" for detailed help of a specific command.
```

A.2.4.2 Specifying Help on a Category of Commands

```
$ abctl help catalog

Usage: abctl command [options]

Command                Description
-------------------------------------------------------------------------------
addToAssembly           Adds an appliance or assembly to an assembly.
connectEndpoints        Create a new connection between two endpoints.
createAssembly          Creates a new empty assembly.
createAssemblyArchive   Creates an assembly archive.
createExternalResources Creates and connects external resources for
                         unconnected outputs.
createTemplate          Creates an appliance template.
delete                  Deletes an appliance or assembly.
describeCatalog         Lists appliances and assemblies in the catalog.
describeEndpoints       Lists endpoints of an appliance or assembly.
export                  Exports an appliance or assembly to disk.
import                  Imports an appliance or assembly from disk.

Try "abctl help <command>" for detailed help of a specific command.
```

A.2.4.3 Help with a -command parameter specified

```
$ abctl help -command captureFileSets

NAME

captureFileSets

SYNOPSIS

captureFileSets -name String [-remoteHost String] [-remoteUser String] \
[-sudoUser String] [-remoteWorkingDir Path] [-remoteCleanup] \ [-privateKeyFile
Path] [-quiet] [-force]

DESCRIPTION

Creates file sets for specified appliance or assembly.

OPTIONS

Name: remoteHost
Aliases: rh
Type: String
Required: false
Value description: String
Default value:
Possible values:
Description: Host name or IP address and optional SSH port of the remote
    machine.  If specified, the remoteUser must be specified as well.

Name: remoteUser
Aliases: ru
Type: String
Required: false
Value description: String
Default value:
```

```
Possible values:
Description: Name of the SSH user to use for accessing the remote machine.  If
    specified, the remoteHost must be specified as well.

...

EXAMPLES

abctl captureFileSets -name myOhs -force
```

A.2.4.4 Help with a -command parameter specified and -usage flag specified

```
$ abctl help -command captureFileSets -usage

Command usage:

captureFileSets -name String [-remoteHost String] [-remoteUser String] \
[-sudoUser String] [-remoteWorkingDir Path] [-remoteCleanup] \ [-privateKeyFile
Path] [-quiet] [-force]

Try 'abctl help -command captureFileSets' for detailed help of the command.
```

B

Oracle Virtual Assembly Builder Introspection Plug-ins

This appendix describes or provides references to other product documentation for plug-ins for appliances that Oracle Virtual Assembly Builder can introspect. For some products being released in Oracle Fusion Middleware 12c, the product documentation for the component contains the plug-in documentation. This appendix describes those component plug-ins that are not documented in other product documentation.

This appendix contains the following sections:

B.1 Plug-ins Documented in Other Product Documentation

For the details of plug-ins for specific Oracle components, see the product documentation for the component. For more information, see:

- "Oracle Coherence*Web Extension for OVAB" in *Administering HTTP Session Management with Oracle Coherence*Web*.
- "OHS Introspector Plug-in for OVAB" in *Administering Oracle HTTP Server*
- "Using the Oracle SOA Suite Oracle Virtual Assembly Builder Plug-in" in *Oracle® Fusion Middleware Administrating Oracle SOA Suite and Oracle Business Process Management Suite*.
- "Using the Introspection Plug-in for Oracle Virtual Assembly Builder" in *Administering Server Environments for Oracle WebLogic Server*.

B.2 Generic Appliance Plug-in

The generic appliance introspection plug-in allows you to create an appliance that gets configured and deployed using scripts supplied during introspection. The generic appliance introspector plug-in reads and collects the properties of an opaque,

standalone, and self-contained product or application, and captures the set of files that make up the product as specified by the user. The output of the plug-in is an appliance. Use the `introspectGenericProd` command to introspect a generic appliance.

A generic appliance does not make use of product-specific plug-in code to capture configuration or product location--instead a simple appliance is created and a set of user-supplied properties, paths, and scripts are added to it in a generic manner. The set of scripts passed in at creation are executed at deployment to perform the necessary operations.

B.2.1 Requirements

The following requirements apply to generic appliances:

B.2.1.1 Specify File Sets

You must specify a list of directories to be captured. All file and directories underneath those directories will be captured. This capability provides the means by which installation binaries, configuration, and data is captured.

B.2.1.2 Scripts Are Launched As Root

All scripts will be launched as the root user. This provides generic appliance scripts the flexibility of performing operations requiring root privileges or switching to another user as desired.

B.2.2 Resulting Artifact Type

A single appliance.

B.2.3 Generic Appliance Plug-in Introspection Parameters

Table B–1 lists the introspection parameters for the generic appliance:

Table B–1 Generic Appliance Plug-in Introspection Parameters

Parameter	Description
`productRoots`	A required list of one or more colon-separated paths of type `string`. Each path becomes a root of a `FileSet` within the definition of the Appliance. All files within each specified root are captured during an operation. Each path gets mapped to its own filesystem on the VM at deployment.
`propertyFile`	Optional. Absolute path of the properties file. Each property in the specified file becomes a user property in the appliance metadata. If `propertyFile` is specified, then any `properties/` directory is ignored.
`scriptRootDir`	Optional. Path to the root directory where the scripts are obtained. Scripts are located in subdirectories within this root directory according to operation type.

B.2.4 Property File

If you specify the `propertyFile` parameter, you must reference a file that exists on the reference system and is readable (otherwise, a failure results and the appliance is not created).

A property file is a text file containing a list of (name, value) pairs. Each property in the property file is added as a user property into the appliance. At deployment the user properties in the appliance are written back out to a file with the absolute path indicated by the $AB_USERPROPS_FILE environment variable.

A property file must consist of zero or more lines where each line is a property declaration, a comment, or a blank line. More formally, a property file must comply with the following syntax:

Example B–1 Property File Syntax

```
property-file   = *line
line            = prop-decl | comment | blank-line
prop-decl       = name "=" value NL
comment         = *WS "#" *CHAR NL
blank-line      = *WS NL
name            = name-start-char *name-body-char
name-start-char = <any character in "a".."z", "A".."Z", "_">
name-body-char  = <any character in "a".."z", "A".."Z", "0".."9", "_">
value           = *SHCHAR | SQ *SHCHAR SQ | DQ *SHCHAR DQ
NL              = <platform dependent line termination sequence>
WS              = <white space character>
CTL             = <any control character (octets 0 -31) and DEL (127)>
CHAR            = <any character, excluding CTL (and NL), but including WS>
SHCHAR          = <any CHAR, escaped as necessary for shell interpretation>
SQ              = <single quote>
DQ              = <double quote>
```

Any property file that does not comply with the above syntax rules results in an error, and an appliance is not created. Property declarations must be contained on a single line. Ending a line with a slash ("\") does not result in line continuation.

All properties will be marked as "required" in the appliance metadata. Property declarations without any assigned value (nothing after "=") will be set to null in the appliance metadata requiring that the user assign a value to that property prior to deployment.

Whitespace is not permitted anywhere to the left of the equal sign ("=") in a property declaration. Whitespace to the right of the equal sign is assumed to be part of the intended value and is preserved (resulting in a failure if the value is sourced).

Quotes around property values are preserved and are visible to users as part of the value. When editing a property value, it is the responsibility of the user to add, remove, or preserve quotes as necessary according to the rules of shell interpretation.

Comments and blank lines are discarded at dehydration and are not reproduced when the file is regenerated at rehydration.

Typically, a generic appliance script reads the property file into the script environment. A common usage pattern is:

```
#!/bin/bash
#
# This script reconfigures the example server of the
# example product.
#
. $AB_USERPROPS_FILE
$ORACLE_HOME/bin/oim_reconfig.sh $OIM_INSTANCE
```

Here is sample content of a valid properties file:

```
# The following property must have a user supplied value
```

```
SHELL=
# This is a variable that should not be changed
PRODUCT_HOME=/my/install/will/not/move
PRODUCT_INSTANCE=/my/instance/will/also/not/move
PRODUCT_PROPERTY="Hello World"
# This is a mispelled variable name
TRUSTROTE=/path/to/file.jks
```

Given the above property file example (including user edits of some values), the following property file content is generated during reconfiguration:

```
SHELL=/bin/bash
PRODUCT_HOME=/my/install/will/not/move
PRODUCT_INSTANCE=/my/instance/will/also/not/move
PRODUCT_PROPERTY="Yo, peoples of planet Earth!"
TRUSTROTE=/path/to/file.jks
```

B.2.4.1 Script Root Directory

The script root directory is the top level directory containing the script subdirectories. If the specified directory does not exist or is not readable then an error will be returned and an appliance will not be created

User supplied reconfiguration scripts must be placed within the root script directory under the following well-known subdirectories: config.d/, start.d/, ping.d/, stop.d/. Scripts under each subdirectory will be captured during dehydration and stored in the appliance. During rehydration the appropriate set of scripts according to the requested operation will be executed in lexicographical order (same order as /bin/ls).

The following is an example of the set of script directories the user might create:

```
/path/to/script/dirs/
     config.d/
         00config.sh
         01.config.sh
     start.d/
         00start.sh
         01start.sh
     stop.d/
          stop.sh
     ping.d/
          ping.sh
```

Any file or directory located in the script root directory other than the set of well-known subdirectories will be ignored and will not be captured during dehydration.

The script root directory need not contain all well-known subdirectories. The omission of a well-known subdirectory is ignored during dehydration with the assumption that no script is needed for that particular phase.

A well-known subdirectory may be empty. An empty well-known subdirectory will not be captured.

Well-known subdirectories must only contain scripts that should be launched by the generic appliance plug-in. The presence of a directory within a well-known subdirectory will generate an error during dehydration and an appliance will not be created. Everything else will be captured and the generic appliance will attempt to execute it during rehydration. Files such as data files, configuration files, and videos will likely fail to execute properly resulting in an overall failure of the corresponding

operation. Such files are more appropriately captured using the `productRoots` parameters.

B.2.5 Wiring

You can define one or more input and output endpoints to be able to:

- Connect from/to other appliances in the assembly
- Connect to external resources
- Exchange properties between connected appliances: The GenericProd introspector plug-in allows you to specify at introspection a set of directories to capture, a set of properties to expose to the user, and a set of scripts to run during reconfiguration operations. Currently only the set of properties specified during introspection are made available to the specified reconfiguration scripts.

The metadata associated with these inputs and outputs is exposed to your reconfiguration scripts so that the network connections of the underlying captured product are configured according to the details you supply during assembly editing and the environment into which you deploy the appliance.

B.2.5.1 Appliance Inputs

An appliance input represents the configuration of a host and port on which a component opens a socket, waits for connections to be established, and handles incoming requests.

The metadata associated with an appliance input includes the following:

- The port on which the component will establish the socket.
- The host on which the component will establish the socket (determined at deployment).
- The protocol(s) that the component will handle on the socket.
- A set of configurable properties with details about how the endpoint should be configured and/or connected to.
- A set of static properties with details about how the endpoint should be configured and/or connected to.
- The original host and port used by the component at the time of capture.

During reconfiguration, the introspector plug-ins examine the metadata of their own appliance inputs to modify their own configuration of the sockets that it will open for handling incoming requests. Other components that connect to these sockets also examine this metadata during reconfiguration to appropriately modify client-side connection configuration.

B.2.5.2 Appliance Outputs

An appliance output represents the configuration of a host and port to which a component establishes connections and sends requests. During assembly editing connections between appliance outputs and appliance inputs are configured so that the owner of an appliance output can discover the metadata of the appliance input to which it should connect.

The metadata associated with an appliance output includes the following:

- The protocol that the component will use on the connection.

- A set of configurable properties with details about how the component will establish connections.
- A set of static properties with details about how the component will establish connections.
- A set of properties needed on any compatible input to which this output will connect to.
- The original host and port used by the component at the time of capture.

During reconfiguration, the introspector plug-ins examine the metadata of their own appliance outputs and the appliance inputs to which those outputs are connected. This metadata is used to modify the configuration of the sockets that the component will open for sending requests.

B.2.5.3 Endpoint Directory

The optional endpointDir parameter specifies a directory containing one or more files describing appliance inputs and appliance outputs to be added to the appliance as part of introspection. There is one file per endpoint, named as follows:

- For appliance inputs: `<input-name>.input`
- For appliance outputs: `<output-name>.output`

During reconfiguration of GenericProd appliances, the endpoint files are created for all appliance inputs and appliance outputs of the appliance. In addition, files are created for all appliance inputs of other appliances to which the GenericProd appliance is connected.

The `$AB_ENDPOINT_DIR` environment variable is passed to all reconfiguration scripts. This variable is set to the temporary directory created to hold the endpoint files.

B.2.5.4 Format of Endpoint Files

The *.input/*.output files are comprised of properties expressed as key/value pairs, one per line, with the form of *key=value*. Because the keys are not guaranteed to be valid shell variable names, you cannot source endpoint files to create shell variables.

B.2.5.4.1 Properties of *.input files The following properties can be contained in *.input files:

- `port`: required
- `protocols`: required, comma separated list with no whitespace
- `host`: derived internally, available at reconfiguration
- `userprop.`*`property-name`*: optional, always type="STRING"
- `sysprop.`*`property-name`* : optional, always type="STRING"
- `original-port`: optional
- `original-host`: optional

You must specify required properties in the files supplied at introspection. You should not specify the "host" property, which is derived automatically during reconfiguration. In reconfiguration scripts, use the value of the "host" property to set the listening address when configuring an ApplianceInput and to set the connection address when configuring an ApplianceOutput.

B.2.5.4.2 Properties of *.output files The following properties can be contained in *.output files:

- `protocol`: required
- `connected-input`: derived internally, available at reconfiguration
- `userprop.property-name`: optional, always type="STRING"
- sysprop.*property-name*: optional, always type="STRING"
- `original-port`: optional
- `original-host`: optional
- `conn-userprop.property-name`: optional, for specifying requirements on connected inputs
- `conn-sysprop.property-name`: optional, for specifying requirements on connected inputs

You must specify required properties in the files supplied at introspection. You should not specify the "connected-input" property, which is derived automatically during reconfiguration.

B.2.6 Extensions of the Plug-in

None.

B.2.7 Supported Template Types

The supported template type is Oracle Enterprise Linux (OEL).

B.3 Oracle Database (SIDB) Plug-in

The single-instance Oracle Database introspection plug-in examines a single-instance Oracle Database appliance and captures its metadata.

B.3.1 Versions Supported

This plug-in supports versions 10gR2, 11gR1, and 11gR2.

B.3.2 Oracle Database Introspection Parameters

Table B–2 lists the introspection parameters for Oracle Database:

Table B–2 Oracle Database Plug-in Introspection Parameters

Parameter	Description
asmHome	This parameter is required if ASM is used as the storage type and it is installed in a separate Oracle Home.
dbHome	The ORACLE_HOME of the Oracle RDBMS to be introspected.
oracleSid	The Oracle System ID (SID) of the Oracle RDBMS to be introspected.
shutdownDBOK	This flag needs to be passed to approve the database reboot.
sysDBAUserName	Database account with SYSDBA privileges. This parameter is required only if OS authentication is disabled for the current database.

B.3.3 Oracle Database Introspection Password Parameters

Table B–3 lists the introspection password parameters for Oracle Database. When performing introspection using the `abctl` tool, a prompt is shown to enter values for these parameters. Oracle Virtual Assembly Builder Studio provides password fields for these parameters.

Table B–3 Oracle Database Plug-in Introspection Parameters (Prompted During Introspection)

Parameter	Description
rootPassword	Optional. The rootPassword parameter is required to change the permissions of the ORACLE_HOME files to make capturing of file sets possible.
sysDBAPassword	Optional. Password for sysDBAUserName user. This parameter is required only if OS authentication is disabled for the current database.

B.3.4 Reference System Prerequisites

This introspection plug-in does not support configurations with multiple NICs.

B.3.5 Requirements

The following requirements apply to Oracle Database:

The base system image OS version must match the version of the reference system.

B.3.6 Resulting Artifact Type

A single appliance.

B.3.7 Wiring

An input is created on the SIDB appliance with a default Listener and Port. The protocol of an SIDB input is set to 'jdbc'.

B.3.8 Wiring Properties

The input endpoint has two editable properties - `port` and `description`, and two non-editable properties - `protocol` and `ORACLE_SID`. The `protocol` indicates what sort of output can be connected to the input.

B.3.9 Oracle Database Appliance Properties

Assemblies with an Oracle Database appliance have user properties (Table B–4) and system properties (Table B–5).

Table B–4 Oracle SIDB Plug-in: User Properties

Name	Type	Req'd	Default	Description
asm-password	String	false	none	Password for SYS asm account.
db-account-password	Password	true	none	The password for database accounts SYS, SYSTEM, SYSMAN, and DBSNMP.

Table B–5 Oracle SIDB Plug-in System Properties

Name	Type	Req'd	Default	Description
ASM_BASE	String	false	none	ASM ORACLE_BASE path.
ASM_HOME	String	false	none	ASM ORACLE_HOME path.
ASM_OWNER	String	false	none	The OS user who owns the ASM ORACLE_HOME.
DATA_ASM_ DISCOVERY_ STRING	String	false	none	Path to discover all data asm disks.
DATA_ASM_DISK_ GROUP_NAME	String	false	none	Name of the data asm diskgroup.
DATA_ASM_DISK_ GROUP_ REDUNDANCY	String	false	none	Data asm diskgroup level of redundancy.
DATA_ASM_DISKS	String	false	none	Paths of the disks that belong to the data asm diskgroup.
DATA_STORAGE_ TYPE	String	false	none	Storage type for database data.
DB_BASE	String	false	none	The Oracle database base path.
DB_GROUP	String	false	none	The group of the OS user who owns Oracle home.
DB_HOME	String	false	none	The Oracle database home path.
DB_HOME_NAME	String	false	none	The name of the Oracle Home.
DB_LISTENER_ NAME	String	false	none	Oracle database listener name.
DB_ORACLE_ GROUPS	String	false	none	The OSDBA, OSOPER and OSASM groups.
DB_OWNER	String	false	none	The OS user who owns Oracle home.
DB_USING_ASM	String	false	none	Set to *true* if either of database or recovery files are stored on ASM as per reference system.
DB_VERSION	String	false	none	Version of Oracle database software on reference system.
ORACLE_SID	String	false	none	The Oracle database SID.
ORIGINAL_ GLOBAL_DB_NAME	String	false	none	The database unique name on reference system.
RECOVERY_ASM_ DISCOVERY_ STRING	String	false	none	Path to discover all recovery asm disks.
RECOVERY_ASM_ DISK_GROUP_ NAME	String	false	none	Name of the recovery asm diskgroup.
RECOVERY_ASM_ DISK_GROUP_ REDUNDANCY	String	false	none	Recovery asm diskgroup level of redundancy.

Table B–5 (Cont.) Oracle SIDB Plug-in System Properties

Name	Type	Req'd	Default	Description
RECOVERY_ASM_ DISKS	String	false	none	Paths of the disks that belong to the recovery asm diskgroup.
RECOVERY_ STORAGE_TYPE	String	false	none	Storage type for database recovery.

B.3.10 Extensions of the Plug-in

None.

B.3.11 Supported Template Types

The supported template type is Oracle Enterprise Linux (OEL).

B.4 Oracle Forms and Reports Extensions

The Oracle Forms and Reports introspection extensions extend the functionality of the Oracle WebLogic Server introspection plug-in. These examine Forms and Reports applications and configuration residing in the Forms WebLogic servers, Reports WebLogic servers and Oracle Instance.

B.4.1 Versions Supported

The extensions support introspecting only in the following scenarios:

- Oracle Forms and Reports 11*g* Release 2 (11.1.2.0 or newer).
- Only Oracle Access Manager 11*g* Release 1 (11.1.1.5) as the Identity Management/Access Control Server with WebGate as the access client is supported when Forms and Reports applications are to be protected by single sign-on.
- Oracle Access Manager with `mod_osso` as the access client or Oracle Single Sign-On Server with `mod_osso` as the access client is not supported.

B.4.2 Introspection Parameters

There are no additional parameters required beyond those needed by Oracle WebLogic Server.

B.4.3 Reference System Prerequisites

In addition to the reference system prerequisites mentioned for the Oracle WebLogic Server plug-in, create the following empty files on the reference system.

- `$ORACLE_HOME/precomp/public/bnddsc.for`
- `$ORACLE_HOME/precomp/public/oraca.for`
- `$ORACLE_HOME/precomp/public/seldsc.for`
- `$ORACLE_HOME/precomp/public/sqlca.for`

Note: `$ORACLE_HOME` refers to the Forms and Report Oracle Home.

B.4.3.1 Adding Partner Application Registation Utility (Web Tier on a Separate Node)

If the Web tier is set up on a separate node for the Forms and Reports installation on the reference system, copy the partner app registration utility (rreg-toolkit.jar) from the Forms and Reports installation located in the `ORACLE_HOME/oam/server/rreg/client` directory to the `ORACLE_HOME/oam/server/rreg/client` directory on the Web tier before running the introspection on the Web tier node.

B.4.4 Requirements

In addition to the requirements mentioned for Oracle WebLogic Server Plug-in, following requirements must be met:

B.4.4.1 Managed Servers Requirement

Forms and Reports managed servers must be up and running before starting the introspection.

B.4.4.2 Supported Topologies

The Forms and Reports extensions support:

- Only Forms and Reports managed servers (standalone or part of the cluster) that are on the same machine as the Admin Server will be examined and captured.
- In case of an Expand Cluster configuration scenario, on the reference system if there are multiple managed servers in the cluster_forms cluster, the configuration from the WLS_FORMS managed server will be replicated to all the Forms managed servers in the virtual deployed environment. Similarly, the configuration from the WLS_REPORTS managed server will be replicated to all the Reports managed servers in the virtual deployed environment.

B.4.4.3 Unsupported Topology

The Forms and Reports extensions do not support:

- Introspection of remote Forms and Reports managed servers: servers that are created on a different machine than the Admin Server machine.
- Forms and Reports managed servers created through the Remote Extend Domain configuration scenario: the case where the domain pre-exists and the Forms and/or Reports managed servers are added later through the Extend Domain configuration scenario on a different machine than the Admin Server machine.

B.4.4.4 Requirement to Support Scale Out of Deployed Assembly

After deploying the assembly, you can add new managed servers to Forms and Reports WLS clusters through the "scale" operation. But a cluster can only be scaled out up to the "max" scalability property which is limited to the number of managed servers that were present in the cluster in the reference system at introspection time. To account for future scale out, in the reference system, you should temporarily add additional managed servers to the Forms cluster (cluster_forms) and Reports cluster (cluster_reports) using the WLS Admin Console or WLST before that WLS domain is introspected.

These additional managed servers need not be assigned Machines nor do they need to be up and running in the reference system. Once the assembly is created you can remove these temporary managed servers from the reference system. You can control the actual number of Forms and Reports managed servers that have to be deployed

using the "target" scalability property of the cluster_forms WLS appliance. Refer to Table B–19, " Scalability Properties of the Instance Appliance", for details on the scalability properties.

B.4.4.5 Oracle HTTP Server to Reports Cluster Configuration Requirement

In the reference system, if the Reports managed server(s) is part of a WebLogic cluster (cluster_reports) and it's front ended by an Oracle HTTP Server, make sure WebLogicCluster directive is used to for Oracle HTTP Server to Reports managed server(s) routing. By default, it's configured with WebLogicHost and WebLogicPort directives. Modify the following file to add an entry for WebLogicCluster directive and comment out WebLogicHost and WebLogicPort directives:

`$ORACLE_INSTANCE/config/OHS/<ohs_name>/moduleconf/reports_ohs.conf`

For example:

```
#mod_weblogic related entry
#<IfModule mod_weblogic.c>
<Location /reports>
SetHandler weblogic-handler

# Add this line:
WebLogicCluster machine1.domain:port1,machine2.domain:port2

# Comment following two entries
# WebLogicHost machine1.domain
# WebLogicPort port1
</Location>
#</IfModule>
```

B.4.4.6 tnsnames.ora

The `tnsnames.ora` file on the reference system is included in any assemblies that are created, and any databases (and host machines for them) which are referred to in the file are also referred to in the `tnsnames.ora` that is deployed as part of the deployed assemblies. Thus, generally the `tnsnames.ora` file should be empty (or be removed) from the reference system before the assemblies are created, particularly in cases where the assemblies will be shipped to third parties (since those databases and machines will not exist in the new environment). In those cases, users should add their required database entries to the file on the Forms and Reports virtual nodes after deployment.

B.4.5 Resulting Artifact Type

For each Forms and Reports clusters and standalone (non-clustered) managed servers in the introspected WebLogic Server domain, the Forms and Reports extensions create a new appliance within the atomic Oracle WebLogic Server Assembly.

B.4.6 Wiring

The appliances are wired as follows:

B.4.6.1 Oracle HTTP Server Appliance to Forms and Reports WLS Appliances Wiring

If Oracle HTTP Server is configured on the reference system, one or more appliance outputs with the prefix "wls-" are created on the Oracle HTTP Server appliance.

Connect the appliance output with the description property "loc=/forms" to the Forms appliance in the atomic WLS assembly and similarly connect the appliance output with the description property "loc=/reports" to the Reports appliance in the atomic WLS assembly.

B.4.6.2 Forms and Reports WLS Appliances to Oracle Internet Directory External Resource Wiring

On the reference system, if the Forms and Reports managed servers are configured with the Oracle Internet Directory server to support running Forms and Reports applications with single sign on protection, appliance outputs named "OIDConnection" are created on the Forms and Reports appliances.

Create an External Resource for this appliance output and enter the properties described in Section B.4.7, "Wiring Properties". If Forms and Reports are part of the same assembly, use the same External Resource to represent the Oracle Internet Directory Server.

Also refer to Oracle HTTP Server - OAM Server wiring as described in "OHS Introspector Plug-in for OVAB" in *Administering Oracle HTTP Server*.

B.4.6.3 Reports Appliance to Oracle Database Wiring

If a Reports WLS cluster was configured in the reference system, an appliance output named "job_repos_db" is created on the Reports cluster appliance (cluster_reports). This output should be connected to an Oracle Database appliance or an External Resource representing an Oracle Database that contains required Reports job database schemas (see Oracle Reports documentation for details about such schemas) and enter the properties defined in Section B.4.7, "Wiring Properties".

B.4.7 Wiring Properties

The following properties should be set for the External Resource (refer to Section B.4.6.2, "Forms and Reports WLS Appliances to Oracle Internet Directory External Resource Wiring") representing the Oracle Internet Directory server.

Table B–6 Oracle Internet Directory External Resource Properties

Name	Type	Req'd	Default	Description
UseOID	Boolean	true	true	This property determines whether or not to configure Forms-OID wiring in the virtual deployment. This property is set to *true* if the reference system was setup with Forms-OID/Reports-OID configuration. Setting this property to *false* disables Forms-OID/Reports-OID configuration in the virtual deployments. When this property is set to *true*, set the rest of the properties listed in this table.
Host	String	true	none	OID Server host name.
Port	Integer	true	none	OID Server port number.
OID_UserDn	Boolean	true	none	OID Server Administrator user name (typically `orcladmin`).
OID_Password	String	true	none	OID Server Administrator password.

Table B–6 (Cont.) Oracle Internet Directory External Resource Properties

Name	Type	Req'd	Default	Description
sslConnection	Boolean	true	false	This property indicates whether to establish an SSL connection with the OID Server. When this property is set to *true*, you should provide the OID Server's SSL port number in the Port property.

Note: When Single-Sign-On (SSO) protection is to be enabled for the default Forms and Reports applications in the deployed environment, add the values **/reports/rwservlet/*,/forms/frmservlet?*oamMode=true*** to the `webgate.protectedResourcesList` user property on the OHS Appliance as described in Table B–15.

Note: If the `UseOID` property described in Table B–6 is set to false, that is, you chose not to enable Sign-Sign-On protection on the Forms and Reports applications in the deployed instance, then make sure that the values **/reports/rwservlet/*,/forms/frmservlet?*oamMode=true*** are removed from to the `webgate.protectedResourcesList` user property on the OHS Appliance, as described in Table B–15.

Applicable only for a Reports appliance, the following properties should be set for the Oracle Database appliance or an External Resource representing an Oracle Database that contains required Reports job database schemas.

Table B–7 Oracle Database Appliance Properties

Name	Type	Req'd	Default	Description
global-db-name	String	true	none	The Oracle System ID (SID) of the Oracle Database. Property defined at input endpoint.
port	Integer	true	none	Oracle Database listener port number. Property defined at input endpoint.

Table B–8 Oracle Database External Resource Properties

Name	Type	Req'd	Default	Description
hostname	String	true	none	Oracle Database host name. Property defined at appliance level.
global-db-name	String	true	none	The Oracle System ID (SID) of the Oracle Database. Property defined at input endpoint.
port	String	true	none	Oracle Database listener port number. Property defined at input endpoint.

Table B–9 Reports Appliance Output Properties: job_repos_db

Name	Type	Req'd	Default	Description
username	String	true	none	The user needed for the database connection.
password	String	true	<empty>	The password for the user needed for the database connection.
address-name	String	true	none	The database address/service name that goes in the `tnsnames.ora` file (for example, `myProdDb`).

B.4.8 Oracle Forms and Reports Appliance Properties

Same as for Oracle WebLogic Server appliances.

(Reports only) If the reference system is enabled for a Reports cluster, specify the following property in your deployment plan:

```
wlsAssembly -->FileSets -> reportsClusterSharedDir
```

This is the shared NFS directory to which the VM should mount, and the location of the shared directory for the Oracle Reports shared cache.

B.5 Oracle RAC Database (RACDB) Plug-in

The RACDB introspection plug-in examines Oracle Clusterware and RAC Database components and captures their metadata.

B.5.1 Versions Supported

This plug-in supports version 11gR2.

B.5.2 Oracle RAC Database Introspection Parameters

Table B–10 lists the introspection parameters for the RACDB introspection plug-in:

Table B–10 Oracle RACDB Plug-in Introspection Parameters

Parameter	Description
asmHome	This parameter is required if ASM is used as the storage type and it is installed in a separate Oracle Home.
crsHome	The ORACLE_HOME of the Oracle CRS to be introspected.
dbHome	The ORACLE_HOME of the Oracle RDBMS to be introspected.
globalDbName	The global database name of the Oracle RDBMS to be introspected.
shutdownDBOK	This flag needs to be passed to approve the database reboot.
sysDBAUserName	Optional. Database account with SYSDBA privileges. This parameter is required only if OS authentication is disabled for the current database.

B.5.3 Oracle RAC Database Introspection Password Parameters

Table B–11 lists the introspection password parameters for the Oracle RAC Database plug-in. When performing introspection using the `abctl` tool, a prompt is shown to enter values for these parameters. Oracle Virtual Assembly Builder Studio provides password fields for these parameters.

Table B–11 Oracle RACDB Plug-in Introspection Parameters (Prompted During Introspection)

Parameter	Description
rootPassword	Optional. The rootPassword parameter is required to change the permissions of the ORACLE_HOME files to make capturing of file sets possible.
sysDBAPassword	Optional. Password for sysDBAUserName user. This parameter is required only if OS authentication is disabled for the current database.

B.5.4 Reference System Prerequisites

The Clusterware stack must be running during introspection.

B.5.5 Requirements

The following requirements apply to Oracle RAC Database:

The base system image OS version must match the version of the reference system.

B.5.6 Resulting Artifact Type

A single appliance.

B.5.7 Wiring

For an 11.2 series RACDB, a single input is created on the RACDB appliance.

For a pre-11.2 series RACDB, inputs are created on the RACDB appliance for each Listener or Port directive found in the configuration.

B.5.8 Wiring Properties

For 11.2 series RACDB, the input endpoint has two editable properties, `scan-name` and `global-db-name`, and two non-editable properties, `protocol` and `port`, which indicate what sort of output can be connected to the input.

For pre-11.2 series RACDB, the input endpoint has a editable property `global-db-name` and two non-editable properties, `protocol` and `port`.

B.5.9 Oracle Database Appliance Properties

Assemblies with an Oracle Database appliance have user properties (Table B–12) and system properties (Table B–13).

Table B–12 Oracle RAC Database: User Properties

Name	Type	Req'd	Default	Description
asm-password	String	false	none	Password for SYS asm account.
cluster-name	String	false	new_cluster	Name for cluster (only for pre-11.2 series Oracle Database).
db-account-password	Password	true	none	The password for database accounts SYS, SYSTEM, SYSMAN, and DBSNMP.

Table B–13 Oracle RAC Database System Properties

Name	Type	Req'd	Default	Description
CRS_BASE	String	false	none	The Clusterware base path.
CRS_HOME	String	false	none	The Clusterware home path.
CRS_OWNER	String	false	grid	The name of the OS user who will be the owner of the Clusterware home.
CRS_GROUP	String	false	oinstall	The name of the OS user group of the owner of the Clusterware home.
CRS_ORACLE_ GROUPS	String	false	oinstall	The OSDBA, OSOPER and OSASM groups.
VOTING_DISKS_ LOCATIONS	String	false	none	Locations of voting disks (only for File System storage type for clusterware files).
VOTING_DISKS_ REDUNDANCY	String	false	none	Voting disks redundancy. (only for File System storage type for clusterware files).
OCR_DISKS_ LOCATIONS	String	false	none	Locations of ocr disks(only for File System storage type for clusterware files).
OCR_DISKS_ REDUNDANCY	String	false	none	OCR disks redundancy. (only for File System storage type for clusterware files)
SCAN_PORT	String	false	1521	Port for SCAN listener.
CRS_STORAGE_ TYPE	String	false	none	Storage type for clusterware files as per reference system.
CRS_VERSION	String	false	none	Version of Clusterware software on reference system.
CRS_ASM_DISK_ GROUP_NAME	String	false	OVMOCRVD	Name of the clusterware asm diskgroup.
CRS_ASM_ DISCOVERY_ STRING	String	false	/dev/raw/ovmocrvd*	Path to discover all data asm disks.
CRS_ASM_DISK_ GROUP_ REDUNDANCY	String	false	NORMAL	Clusterware asm diskgroup level of redundancy.
CRS_ASM_DISKS	String	false	/dev/raw/ovmocrvd0,/dev/raw/ovmocrvd1,/dev/raw/ovmocrvd2	Paths of the disks that belong to clusterware asm diskgroup.
ASM_BASE	String	false	none	ASM ORACLE_BASE path.
ASM_HOME	String	false	none	ASM ORACLE_HOME path.
ASM_OWNER	String	false	grid	The OS user who owns ASM ORACLE_HOME.
DATA_ASM_ DISCOVERY_ STRING	String	false	/dev/raw/asm*	Path to discover all data asm disks.
DATA_ASM_DISK_ GROUP_NAME	String	false	none	Name of the data asm diskgroup.

Table B–13 (Cont.) Oracle RAC Database System Properties

Name	Type	Req'd	Default	Description
DATA_ASM_DISK_GROUP_REDUNDANCY	String	false	none	Data asm diskgroup level of redundancy.
DATA_ASM_DISKS	String	false	none	Paths of the disks that belong to data asm diskgroup.
RECOVERY_ASM_DISCOVERY_STRING	String	false	/dev/raw/asm*	Path to discover all recovery asm disks.
RECOVERY_ASM_DISK_GROUP_NAME	String	false	none	Name of the recovery asm diskgroup.
RECOVERY_ASM_DISK_GROUP_REDUNDANCY	String	false	none	Recovery asm diskgroup level of redundancy.
RECOVERY_ASM_DISKS	String	false	none	Paths of the disks that belong to the recovery asm diskgroup.
RECOVERY_STORAGE_TYPE	String	false	none	Storage type for database recovery.
DATA_STORAGE_TYPE	String	false	none	Storage type for database as per reference system.
RECOVERY_STORAGE_TYPE	String	false	none	Storage type for database recovery files as per reference system.
DB_USING_ASM	String	false	none	Set to *true* if either of database or recovery files are stored on ASM as per reference system.
DB_VERSION	String	false	none	Version of Oracle database software on reference system.

B.5.10 Extensions of the Plug-in

None.

B.5.11 Supported Template Types

The supported template type is Oracle Enterprise Linux (OEL).

B.6 Oracle Service Bus Support

Support for Oracle Service is provided via Oracle WebLogic Server plug-in. There is no separate plug-in for it. Oracle WebLogic Server plug-in can be used to introspect a Oracle WebLogic Server domain where Oracle Service Bus is configured.

B.6.1 Versions Supported

This plug-in supports version 11gR1 11.1.1.6.

B.6.2 Oracle Service Bus Introspection Parameters

There are no additional parameters required beyond those needed by Oracle WebLogic Server.

B.6.3 Reference System Prerequisites

There are no additional prerequisites beyond those defined by Oracle WebLogic Server.

B.6.4 Requirements

The following requirements apply to an Oracle WebLogic Server domain containing Oracle Service Bus.

B.6.4.1 Supported Domains

Supported OSB domains are where there is single server, one managed server with Oracle Service Bus and SOA or separate Oracle Service Bus and SOA clusters.

B.6.4.2 Requirement for Configuration Archiving

Turn on configuration archiving since domain security configurations will have to be modified to facilitate the introspection, it is advisable to turn on the configuration backup in WebLogic Server. Refer to the WebLogic Server documentation on configuration backup.

B.6.4.3 Export and Optionally Delete the OSB Artifacts from the Reference Domain

It is recommended that the OSB artifacts be exported before introspection. Optionally they can be deleted.

B.6.4.4 Delete Temporary Files from the Domain Directory

Before introspection and capturing file sets of the domain, all temporary files under the domain directory can be cleaned up. OVAB uses WebLogic pack utility to archive the domain directory from the reference domain. Files containing '\' in their names cannot be processed by the pack utility. They have to be manually removed before introspection.

B.6.4.5 Post Assembly Deployment Requirements

The following are requirements after deployment.

B.6.4.5.1 Import the OSB Artifacts

After the successful deployment of the OSB assembly instance, you can import the OSB artifacts that were exported and optionally deleted before the assembly creation began.

Create a new session for configuration changes and import the OSB artifacts that were saved earlier. After the successful import of the OSB artifacts, the session can be activated.

B.6.4.5.2 Apply Customizations

Some of the endpoint URLs may have to be customized in the deployed VMs as the network configuration is different. You can create a customization file for the assembly instance and after making necessary changes, execute the customizations.

Create a new session for configuration changes and apply the customizations using the OSB console. After execution of the customizations, the change session can be activated.

B.6.5 Resulting Artifact Type

Same as in Oracle WebLogic Server plug-in.

B.6.6 Wiring

Same as in Oracle WebLogic Server plug-in.

B.6.7 Wiring Properties

Same as in Oracle WebLogic Server plug-in.

B.6.8 Oracle Service Bus Appliance Properties

Same as in Oracle WebLogic Server plug-in.

B.6.9 Supported Template Types

The supported template type is Oracle Enterprise Linux (OEL).

B.7 Oracle Traffic Director Plug-In

The Oracle Traffic Director plug-in introspects an Oracle Traffic Director *configuration* on a reference system. In the context of Oracle Traffic Director, a configuration is a collection of configurable elements (metadata) that determine the run-time behavior of an Oracle Traffic Director instance.

For an overview of Oracle Traffic Director, see "Getting Started with Oracle Traffic Director" in the *Oracle Traffic Director Administrator's Guide*.

B.7.1 Versions Supported

The Oracle Traffic Director plug-in supports Oracle Traffic Director 11.1.1.6.

B.7.2 Oracle Traffic Director Introspection Parameters

Table B–14 lists the introspection parameters for Oracle Traffic Director. All the parameters are mandatory.

Table B–14 Oracle Traffic Director Plug-In Introspection Parameters

Parameter	Description
oracleHome	The fully qualified path to the directory in the reference system in which the Oracle Traffic Director binaries are installed.
oracleInstance	The fully qualified path to the directory in the reference system in which the Oracle Traffic Director administration server instance exists.
configName	The name of the Oracle Traffic Director configuration that the plug-in should introspect.

B.7.3 Reference System Prerequisites

For the Oracle Traffic Director plug-in to successfully introspect the Oracle home and the administration server instance, the following prerequisites must be fulfilled on the reference system:

- Oracle Traffic Director must be installed.
- The Oracle Traffic Director administration server instance must be configured.
- At least one configuration must be available.

> **Note:** The administration server need not be running.

B.7.4 Resulting Artifact Type

The result of the introspection is an atomic assembly that contains the following:

- An Oracle Traffic Director administration server appliance
- An Oracle Traffic Director instance appliance

  By default, when the instance appliance is deployed, two virtual machines will be created, regardless of the number of nodes to which the Oracle Traffic Director configuration was deployed on the reference system. The number of virtual machines to be created during the deployment process can be controlled through the scalability properties (see Table B–17, " Scalability Properties of the Administration Server Appliance").

B.7.5 Wiring

The appliances in the Oracle Traffic Director assembly can be wired to external components, and to other appliances and assemblies, through input and output endpoints.

> **Note:** In this release, you can wire the Oracle Traffic Director assembly to only a WLS assembly.

B.7.5.1 Wiring Endpoints of the Administration Server Appliance

An input endpoint is created for the HTTPS listen port of the Oracle Traffic Director administration server appliance. The input endpoint of the administration server appliance has two editable properties, `port` and `description`.

No output endpoint is created for the administration server appliance.

B.7.5.2 Wiring Endpoints of the Instance Appliance

An input endpoint is created for each HTTP listen port in the Oracle Traffic Director configuration. Each input endpoint of the instance appliance has two editable properties, `port` and `description`.

An output endpoint is created for each origin-server pool in the configuration. Each output endpoint of the instance appliance has one editable property, `description`.

Note: Details of the origin servers in the origin-server pool are not captured during introspection. The origin servers will be defined during the deployment process, based on the appliance to which the output endpoint of the Oracle Traffic Director instance appliance is wired.

B.7.6 Oracle Traffic Director Appliance Properties

This section describes the editable properties of the appliances in an Oracle Traffic Director assembly. The properties are captured during introspection and are used to create the virtual machines when the appliances are deployed.

This section contains the following subsections:

- Section B.7.6.1, "Editable Properties of the Administration Server Appliance"
- Section B.7.6.2, "Editable Properties of the Instance Appliance"

B.7.6.1 Editable Properties of the Administration Server Appliance

Table B–15 General Properties of the Administration Server Appliance

Name	Type	Req'd	Default	Description
adminUser	String	true	`admin`	The user name for logging in to the administration server.
adminPassword	String	true	None	The password for the administration server user name.

Table B–16 Resource Properties of the Administration Server Appliance

Property Name	Type	Req'd	Default Value	Description
CPU_MHZ	Integer	true	1000	CPU clock speed.
MEMORY_MB	Integer	true	1024	Memory requirement, in megabytes.
NUMBER_CPUS	Integer	true	1	Number of processors.

Table B–17 Scalability Properties of the Administration Server Appliance

Property Name	Type	Req'd	Default Value	Description
Min	Integer	true	1	The minimum number of virtual machines of the appliance that can be deployed.
Max	Integer	true	1	The maximum number of virtual machines of the appliance that can be deployed. This value must be higher than the Absolute-Max value.
Absolute-Max	Integer	false	1	The absolute maximum number of virtual machines of the appliance that can be deployed.
Target	Integer	true	1	The actual number of virtual machines of the appliance to be deployed.

You can change the values of the scalability properties, but Oracle recommends that you leave them at the default values.

B.7.6.2 Editable Properties of the Instance Appliance

Table B–18 Resource Properties of the Instance Appliance

Property Name	Type	Req'd	Default Value	Description
CPU_MHZ	Integer	true	1000	CPU clock speed.
MEMORY_MB	Integer	true	1024	Memory requirement, in megabytes.
NUMBER_CPUS	Integer	true	1	Number of processors.

Table B–19 Scalability Properties of the Instance Appliance

Property Name	Type	Req'd	Default Value	Description
Min	Integer	true	1	The minimum number of virtual machines of the appliance that can be deployed.
Max	Integer	true	2	The maximum number of virtual machines of the appliance that can be deployed. This value must not be higher than the Absolute-Max value.
Absolute-Max	Integer	false	2	The absolute maximum number of virtual machines of the appliance that can be deployed.
Target	Integer	true	2	The actual number of virtual machines of the appliance to be deployed.

B.7.7 Supported Template Types

Oracle Enterprise Linux.

B.7.8 Post-Deployment Tasks

After deploying the Oracle Traffic Director assembly, you should perform the following tasks:

- In the reference system, if Oracle Traffic Director is configured as the SSL termination point, then it is assumed that the certificates in the certificate database of the configuration belong to the virtual servers and not to the host name of the reference system. If the certificates belong to the host name of the reference system, then those certificates should be re-installed on the target virtual machines.
- If SSL is configured between Oracle Traffic Director and the Oracle WebLogic Server managed servers, any new certificates, if required, should be installed after deploying the Oracle Traffic Director assembly.
- Failover groups configured in the reference system are not captured during introspection. Therefore, high availability should be configured afresh by using the Oracle Traffic Director administration console or the CLI.

Note: High availability is supported only when your base image includes *keepalived*.

B.8 Oracle Tuxedo Plug-In

The Oracle Tuxedo introspection plug-in examines a single or multiple-machine Oracle Tuxedo domain, and the Oracle Home Directory that it resides on. The Oracle Home Directory where Tuxedo is installed can also include the Tuxedo add-ons listed below, and those will also be examined:

- Oracle TSAM
- Oracle SALT
- Oracle Tuxedo Application Runtime for CICS and Batch
- Oracle Tuxedo Mainframe Adapter SNA

A single machine domain and its Home Directory, including add-on products, are captured. For a multiple-machine Oracle Tuxedo domain, each machine must be introspected separately and wired into an assembly. See Section B.8.6, "Wiring".

B.8.1 Versions Supported

This plug-in supports version 11gR1.

B.8.2 Oracle Tuxedo Introspection Parameters

Table B–20 lists the introspection parameters for the Oracle Tuxedo introspection plug-in:

Table B–20 Oracle Tuxedo Plug-in Introspection Parameters

Parameter	Description
TUXDIR	Location where Oracle Tuxedo is installed.
TUXCONFIG	Location of the application configuration file, in compiled form. This contains the Tuxedo core configuration as well as a minimal set or values (APPDIR, etc.).
environmentScript	This script will be run before doing introspection to set the environment of the Tuxedo application. The script will be searched relative to the `$APPDIR` directory. As a result of running the script only known Tuxedo-related environment variables will be captured, so as to prevent things such as DISPLAY, or SHELL from being captured that could interfere on the target environment. If not set, the plug-in will attempt to run a `setenv.sh` script (with that exact name) from the `$APPDIR` directory. This behavior will be exclusive, that is, only one script will be run at most.The plug-in will use the output of the `env` command, so care should be taken that such environment setting scripts' output may not interfere with the result of calling `env`. If an environment script is not used, or is used but non-Tuxedo environment variables also need to be set, a well-known standard Java properties file named `ovab-application.properties` will be searched for in `$APPDIR`.
oracleClientDir	Location of Oracle Database client software installation, typically the directory in which the Oracle Instant client software has been unzipped. It is the user's responsibility to ensure that this is accurate, as the Tuxedo plug-in does not have the means to verify that this installation is valid.

Table B–20 (Cont.) Oracle Tuxedo Plug-in Introspection Parameters

Parameter	Description
tnsNamesLocation	Location of the `TNSNAMES.ora` client configuration file. This file is parsed and made a template which results in an Appliance Output being created if either the database name specified in the `ubbconfig OPENINFO` string or environment variable `ORACLE_SID` (in that order or priority) is found. Multiple `OPENINFO/TNSNAMES.ora` entries will result in multiple outputs being generated.
scriptWorkingDir	The working directory where the environment script will be run from. This is useful when scripts use the current working directory to determine path values.
artSecurityProfile	Location of security profile used by ART Batch or ART CICS.

B.8.3 Reference System Prerequisites

None.

B.8.4 Requirements

The following requirements apply to Oracle Tuxedo:

B.8.4.1 Base Image Requirements

The base system image OS version must match the version of the reference system.

Additionally, you must set IPC kernel parameters on the base system image according to the guidelines listed in *Oracle Tuxedo: Installing the Oracle Tuxedo System*: `http://docs.oracle.com/cd/E18050_01/tuxedo/docs11gr1/install/insappd.html`).

B.8.4.2 ART CICS/Batch Applications Requiring Microfocus or COBOL IT

For ART CICS/Batch applications which require Microfocus or COBOL IT to be installed, you must create a new base image with Microfocus or COBOL IT pre-installed (the installation path is the same as it is on the reference system) based on the original Oracle Virtual Assembly Builder base image, and then use the new base image to create template for the ART CICS/Batch application.

Only by following this configuration procedure will ART CICS/Batch applications which require Microfocus/COBOL IT boot successfully on the deployed VM.

B.8.4.3 Requirements Related to Scaling

For TMA SNA, scaling is not applicable.

For ART Batch, scaling is applicable, except for one limitation: if the TMQUEUE server which monitors JES2QSPACE queue space runs on a slave machine, you should not use the scaling feature for that machine for ART Batch.

For ART CICS, not all servers are applicable for scaling. Refer to the ART CICS reference guide to determine whether scaling is applicable or not for specified servers of ART CICS.

B.8.5 Resulting Artifact Type

The resulting artifact type depends on whether you introspect a single-machine or multi-machine domain.

B.8.5.1 Single-Machine Oracle Tuxedo Domain

A single scalable appliance for a single-machine Tuxedo domain.

B.8.5.2 Multi-Machine Oracle Tuxedo Domain

For multi-machine Tuxedo domains, each machine in the reference system must be introspected separately. The resulting appliances are of the following types:

- Master: single non-scalable appliance representing the MASTER node in the Tuxedo domain.
- Backup master (optionally): single non-scalable appliance representing the BACKUP MASTER node in the Tuxedo domain. Must be introspected if present on the reference system.
- Other: single scalable appliance representing a non-master and non-backup node in the reference Tuxedo domain. There can be one or many such types of appliances depending on the topology of the reference system.

To deploy the domain, an empty assembly must be created manually, or the appliances must be included in an existing assembly and the wiring performed.

B.8.6 Wiring

This section describes wiring.

B.8.6.1 Multi-Machine Wiring

Inputs will be created on a Master appliance for each machine (except itself) present in the reference system. These are required for non-Master appliances to obtain information on the Master appliance at rehydration time.

Outputs will be created on a Master appliance for each machine (except itself) present in the reference system. These are required for the Master appliance to obtain information on the non-Master appliances at rehydration time. Corresponding inputs and outputs will also be created on non-Master appliances.

These outputs must all be connected to an appliance before deployment. The name of the output and the protocol supported by the output will give hints about the type of appliance to connect the output to.

B.8.6.2 Other Inputs and Outputs

Inputs will be created on an Oracle Tuxedo appliance for the following types of configuration found:

- WSL (Oracle Tuxedo WorkStation protocol)
- JSL (Oracle Jolt)
- ISL (Oracle Tuxedo IIOP protocol)
- Domain (Oracle Tuxedo Domain Gateway)

Outputs will be created on an Oracle Tuxedo appliance for the following types of configuration found:

- Domain (Oracle Tuxedo Domain Gateway)
- Oracle Single-Instance Database
- TMA_SNA (Oracle Tuxedo Mainframe Adapter SNA)
- TSAM (Oracle TSAM)

These outputs must all be connected to either an external resource or to an appliance before deployment. The description on the output and the protocol supported by the output will give hints about the type of appliance to connect the output to.

B.8.7 Wiring Properties

All input endpoints have two editable properties - `port` and `description`, and one non-editable property - a list of `protocols`. The protocols indicate what sort of outputs can be connected to the input.

All output endpoints have one editable property - `description`, and two non-editable properties - `protocol` and `singleton`. The protocol indicates what sort of input can be connected to the output. Singleton indicates what sort of appliance the output can be connected to. If singleton is true, the output can only be connected to an input on an appliance that has a scalability absolute max value of 1.

The following properties are specific to Oracle Tuxedo endpoints (Table B–21 through Table B–23):

Table B–21 Oracle Tuxedo: Appliance Output Properties: Domain

Name	Type	Req'd	Default	Description
existing-address	String	false	Address of the remote domain from the reference system.	Specifies the address of the remote domain this domain will connect to. Only used if the output is connected to an external resource.

The output for TMA_SNA and TSAM can only be connected to an external resource.

Table B–22 Oracle Tuxedo: Appliance Output Properties: TMA_SNA

Name	Type	Req'd	Default	Description
tma-sna-crm-host	String	false	IP address of remote CRM Server from reference system.	Specifies the IP address of the remote CRM Server this machine will connect to. Only used if the output is connected to an external resource.
tma-sna-crm-port	String	false	Port of remote CRM Server from reference system.	Specifies the port of the remote CRM Server this machine will connect to. Only used if the output is connected to an external resource.
tma-sna-crm-address	String	false	Hex format IP Address and port of remote CRM Server from reference system.	Specifies Hex format IP Address and port of remote CRM Server this machine will connect to. Only used if the output is connected to an external resource.

Table B–23 Oracle Tuxedo: Appliance Output Properties: TSAM

Name	Type	Req'd	Default	Description
tsam-manager-addr	String	false	IP Address of remote TSAM manager from reference system.	Specifies the IP Address of remote TSAM manager this machine will connect to. Only used if the output is connected to an external resource.
tsam-manager-port	String	false	Port of remote TSAM manager from reference system	Specifies the port of remote TSAM manager this machine will connect to. Only used if the output is connected to an external resource.

B.8.8 Oracle Tuxedo Appliance Properties

Oracle Tuxedo appliances have user properties (Table B–24) and system properties (Table B–25).

Table B–24 Oracle Tuxedo: User Properties

Name	Type	Req'd	Default	Description
ALOGPFX	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
ALOGRTNSIZE	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
ALTCC	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
ALTCCFLAGS	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
applicationEnvVars	String	false	none	Applications can use this property to specify non-Tuxedo variables using comma-separated keyword/value pairs. For example: CURRENCY=dollar,GROUPNAME=stdev,JDK=/my/jdk/path. This property is populated by the ovab-application.properties file, if it exists in the `$APPDIR` directory.
applicationPassword	String	false	none	If the Tuxedo application uses security (that is, `*RESOURCES` is set to `APP_PW`, `USER_AUTH`, `ACL` or `MANDATORY_ACL`) then this user property must be set to capture the new password to be used at reconfiguration time.

Table B–24 (Cont.) Oracle Tuxedo: User Properties

Name	Type	Req'd	Default	Description
COBCPY	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
COBDIR	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
COBOPT	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
dbPassword	String	false	none	Replacement value for database username when Tuxedo application has an OPENINFO set for Oracle databases (RM type of Oracle_XA in OPENINFO.) For example: For the following OPENINFO value:"Oracle_XA: Oracle_ XA+Acc=P/Scott/*****+SesTm=30+S qlNet=instance1", the *dbPassword* property may be set in which case it is used to regenerate a new encrypted password.
dbUsername	String	false	none	Replacement value for database username when Tuxedo application has an OPENINFO set for Oracle databases (RM type of Oracle_XA in OPENINFO.) For example: For the following OPENINFO value:"Oracle_XA: Oracle_ XA+Acc=P/Scott/*****+SesTm=30+S qlNet=instance1"the *dbUsername* property may be set to change "Scott" into a different value for the target machine.
FIELDTBLS	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
FIELDTBLS32	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
FLDTBLDIR	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
FLDTBLDIR32	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.

Table B–24 (Cont.) Oracle Tuxedo: User Properties

Name	Type	Req'd	Default	Description
FSCONFIG	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
FSMAXCOMMIT	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
FSMAXUPDATE	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
FSMSGREP	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
FSOFFSET	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
ISSANE	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
QMCONFIG	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
runtimeLoadLibraryPath	String	false	none	Populated with the contents of LD_LIBRARY_PATH after setting the environment. The format of this string is be the same as the actual LD_LIBRARY_PATH to be used on the target system.
shutdownScript	String	false	none	The name of a shutdown script that will be used in place of the `tmshutdown -y` command used on the target machine when it is stopped (after undeployment, or as a result of an Oracle Virtual Assembly Builder `stop` command).
startupScript	String	false	none	The name of a startup script that will be used in place of the `tmboot -y` command used on the target machine when it is started (after deployment or as a result of an Oracle Virtual Assembly Builder `start` command).
TAGENTLOG	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
TM_CBL_IGNORE_CONTEXT	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.

Table B–24 (Cont.) Oracle Tuxedo: User Properties

Name	Type	Req'd	Default	Description
TM_CPAU	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
TM_ENGINE_ TMSHMSEGSZ	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
TM_GWT_ OLDSECCHECK	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
TM_ICU_ COMPATIBILITY	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
TM_LOG_ESYS	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
TM_ORB_ CLTMAXRTY	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
TMCMPLIMIT	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
TMCMPPRFM	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
TMNETLOAD	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
TMNOTHREADS	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
TMSICACHEENTRIESMAX	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
TMUSEIPV6	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
TPMBACONV	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.

Table B–24 (Cont.) Oracle Tuxedo: User Properties

Name	Type	Req'd	Default	Description
TPMBENC	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
TUX_BLOCKLICIW	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
TUX_SSL_ ENFORCECONSTRAINTSUINMEDSIGS	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
URLENTITYCACHEDIR	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
URLENTITYCATCHING	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
VIEWDIR	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
VIEWDIR32	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
VIEWFILES	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
VIEWFILES32	String	false	none	Tuxedo environment variable. See "tuxenv(5)" in *Oracle Tuxedo File Formats, Data Descriptions, MIBs, and System Processes Reference*.
KIX_TS_DIR	String	false	none	Tuxedo ART CICS environment variable. See "CICS Runtime Environment Variables" in *Oracle Tuxedo Application Runtime for CICS Reference Guide*.
KIX_TD_DIR	String	false	none	Tuxedo ART CICS environment variable. See "CICS Runtime Environment Variables" in *Oracle Tuxedo Application Runtime for CICS Reference Guide*.
KIX_TD_QSPACE_DEVICE	String	false	none	Tuxedo ART CICS environment variable. See "CICS Runtime Environment Variables" in *Oracle Tuxedo Application Runtime for CICS Reference Guide*.

Table B–24 (Cont.) Oracle Tuxedo: User Properties

Name	Type	Req'd	Default	Description
KIX_TD_QSPACE_ NAME	String	false	none	Tuxedo ART CICS environment variable. See "CICS Runtime Environment Variables" in *Oracle Tuxedo Application Runtime for CICS Reference Guide*.
KIX_TD_QSPACE_ IPCKEY	String	false	none	Tuxedo ART CICS environment variable. See "CICS Runtime Environment Variables" in *Oracle Tuxedo Application Runtime for CICS Reference Guide*.
KIX_TECH_DIR	String	false	none	Tuxedo ART CICS environment variable. See "CICS Runtime Environment Variables" in *Oracle Tuxedo Application Runtime for CICS Reference Guide*.
KIX_CWA_SIZE	String	false	none	Tuxedo ART CICS environment variable. See "CICS Runtime Environment Variables" in *Oracle Tuxedo Application Runtime for CICS Reference Guide*.
KIX_CWA_IPCKEY	String	false	none	Tuxedo ART CICS environment variable. See "CICS Runtime Environment Variables" in *Oracle Tuxedo Application Runtime for CICS Reference Guide*.
KIX_QSPACE_ IPCKEY	String	false	none	Tuxedo ART CICS environment variable. See "CICS Runtime Environment Variables" in *Oracle Tuxedo Application Runtime for CICS Reference Guide*.
KIX_TRACE_LEVEL	String	false	none	Tuxedo ART CICS environment variable. See "CICS Runtime Environment Variables" in *Oracle Tuxedo Application Runtime for CICS Reference Guide*.
KIX_MAP_PATH	String	false	none	Tuxedo ART CICS environment variable. See "CICS Runtime Environment Variables" in *Oracle Tuxedo Application Runtime for CICS Reference Guide*.
DATA	String	false	none	Tuxedo ART CICS environment variable. See "CICS Runtime Environment Variables" in *Oracle Tuxedo Application Runtime for CICS Reference Guide*.
SPOOL	String	false	none	Tuxedo ART CICS environment variable. See "CICS Runtime Environment Variables" in *Oracle Tuxedo Application Runtime for CICS Reference Guide*.
TMP	String	false	none	Tuxedo ART CICS environment variable. See "CICS Runtime Environment Variables" in *Oracle Tuxedo Application Runtime for CICS Reference Guide*.

Table B–24 (Cont.) Oracle Tuxedo: User Properties

Name	Type	Req'd	Default	Description
PROCLIB	String	false	none	Tuxedo ART CICS environment variable. See "CICS Runtime Environment Variables" in *Oracle Tuxedo Application Runtime for CICS Reference Guide*.
MT_ACC_FILEPATH	String	false	none	Tuxedo ART CICS environment variable. See "CICS Runtime Environment Variables" in *Oracle Tuxedo Application Runtime for CICS Reference Guide*.
MT_DB_LOGIN	String	false	none	Tuxedo ART CICS environment variable. See "CICS Runtime Environment Variables" in *Oracle Tuxedo Application Runtime for CICS Reference Guide*.
MT_LOG	String	false	none	Tuxedo ART CICS environment variable. See "CICS Runtime Environment Variables" in *Oracle Tuxedo Application Runtime for CICS Reference Guide*.
MT_TMP	String	false	none	Tuxedo ART CICS environment variable. See "CICS Runtime Environment Variables" in *Oracle Tuxedo Application Runtime for CICS Reference Guide*.
MT_KSH	String	false	none	Tuxedo ART CICS environment variable. See "CICS Runtime Environment Variables" in *Oracle Tuxedo Application Runtime for CICS Reference Guide*.

Table B–25 Oracle Tuxedo System Properties

Name	Type	Req'd	Default	Description
appdir	String	false	none	Application Directory, location of the Tuxedo application executables and files.
masterTuxconfig	String	false	none	Location of the TUXCONFIG file for the Master machine in a multi-machine domain. This is necessary to perform scale-out operations.
masterTuxdir	String	false	none	Location of TUXDIR file the Master machine in a multi-machine domain. This is necessary to perform scale-out operations.
model	String	false	none	Indicates whether this appliance is a single-machine appliance (SHM) or multi-machine appliance (MP).

Table B–25 (Cont.) Oracle Tuxedo System Properties

Name	Type	Req'd	Default	Description
pmid	String	false	none	Oracle Tuxedo Machine identifier for this appliance.
role	String	false	none	Along with model, is used to qualify the type of appliance when it is part of a multi-machine domain. The role can be 'MASTER', 'BACKUP' or 'SLAVE'. It is always 'SLAVE' for a single-machine domain.
tuxconfig	String	false	none	Used to save the value of TUXCONFIG as introspected.
tuxdir	String	false	none	Used to save the value of TUXDIR as introspected.
kixdir	String	false	none	Used to save the value of KIXDIR as introspected.
kixconfig	String	false	none	Used to save the value of KIXCONFIG as introspected.
jesdir	String	false	none	Used to save the value of JESDIR as introspected.

B.8.9 Extensions of the Plug-in

None.

B.8.10 Supported Template Types

The supported template type is Oracle Enterprise Linux (OEL).

C

Common Properties for Oracle Virtual Assembly Builder Components

This appendix describes common properties for components that Oracle Virtual Assembly Builder can introspect, and other properties that can be specified for deployment. It contains the following sections:

C.1 Common Properties

The following OCM-related properties are common to all appliances.

Table C–1 OCM-related Common Properties

Name	Req'd	Default	Description
ocm.anonymousEmailRegistration.emailId	false	none	Email address to use to register with OCM using an email address that is not associated with a metalink account.
ocm.metalinkCsiRegistration.CSI	false	none	Register deployments using a Customer Support Identifier.
ocm.metalinkCsiRegistration.countryCode	false	none	Two-letter country code associated with the CSI.
ocm.metalinkCsiRegistration.metalinkId	false	none	Metalink ID associated with the CSI.
ocm.metalinkEmailRegistration.metalinkEmailId	false	none	Register deployments using an email ID associated with a metalink account.
ocm.metalinkEmailRegistration.metalinkPassword	false	none	Password associated with the metalink account.
ocm.proxyHost	false	none	Required when OCM registration must occur through a proxy.
ocm.proxyPassword	false	none	Required when OCM registration must occur through a proxy.
ocm.proxyPort	false	none	Required when OCM registration must occur through a proxy.

Table C–1 (Cont.) OCM-related Common Properties

Name	Req'd	Default	Description
ocm.proxyUsername	false	none	Required when OCM registration must occur through a proxy.
ocm.repeaterURI	false	none	For use when registering through a configured OCM hub.
ocm.runConfiguration	false	false	Set to true in order to perform OCM registration.

C.2 System Properties

The following OCM-related system properties are common to all appliances. They cannot be modified by users.

Table C–2 OCM-related System Properties

Name	Req'd	Default	Description
ocm.ccrDirPath.0	true	sample value: /swat/middleware_ps1/utils/ccr	Not to be edited by users.

C.3 External Resource Properties

External resources represent services to which a WebLogic domain connects.

C.3.1 Common Properties

All external resource appliances have the properties described in Table C–3 (`hostname` is a user property and `external-appliance` is a system property).

Table C–3 External Resource Appliance Template Properties: Common Properties

Name	Type	Req'd	Default	Description
hostname	String	false	none	The hostname where the service the external resource appliance is representing resides. By default this value is unset in the external resource appliance templates. You must provide a value before deployment.

C.3.2 foreignJMS Properties

Table C–4 describes properties for external resource appliances to connect a foreign JMS output on an Oracle WebLogic Server Administration Server.

Table C–4 foreignJms Properties

Name	Type	Req'd	Default	Description
url	String	false	none	The URL used to connect to the foreign JMS server. If not specified in the template then the value from the reference system is retained when the WebLogic domain is deployed.

Table C–4 (Cont.) foreignJms Properties

Name	Type	Req'd	Default	Description
Password	String	false	none	The password used to connect to the foreign JMS server. If not specified in the template then the value from the reference system will be retained when the WebLogic domain is deployed.

C.3.3 jmsBridgeDestination Properties

Table C–5 describes properties for external resource appliances to connect a JMS messaging bridge output on an Oracle WebLogic Server Administration Server.

Table C–5 jmsBridgeDestination Properties

Name	Type	Req'd	Default	Description
url	String	false	none	The URL used to connect to the JMS bridge destination server. If not specified in the template then the value from the reference system is retained when the WebLogic domain is deployed.
Username	String	false	none	The username used to connect to the JMS bridge destination server. If not specified in the template then the value from the reference system is retained when the WebLogic domain is deployed.
Password	String	false	none	The password used to connect to the JMS bridge destination server. If not specified in the template then the value from the reference system is retained when the WebLogic domain is deployed.

C.3.4 LDAP Properties

Table C–6 describes properties for external resources to connect an LDAP output on an Oracle WebLogic Server Administration Server.

Table C–6 LDAP Properties

Name	Type	Req'd	Default	Description
Username	String	false	none	The username used to connect to the LDAP server. If not specified in the template then the value from the reference system is retained when the WebLogic domain is deployed.
Password	String	false	none	The password used to connect to the LDAP server. If not specified in the template then the value from the reference system is retained when the WebLogic domain is deployed.

C.3.5 Non-Oracle JDBC Properties

Table C–7 describes properties for external resources to connect a non-Oracle JDBC output on an Oracle WebLogic Server Administration Server.

Table C–7 Non-Oracle JDBC Properties

Name	Type	Req'd	Default	Description
url	String	false	none	The URL used to connect to the non-Oracle database. If not specified in the template then the value from the reference system is retained when the WebLogic domain is deployed.

C.3.6 JDBC Properties

Table C–8 describes properties for external resources to connect an Oracle JDBC output on an Oracle WebLogic Server Administration Server.

Table C–8 JDBC Properties

Name	Type	Req'd	Default	Description
global-db-name	String	false	orcl	The global-db-name needed to connect to the Oracle database. If not specified in the external resource user properties, then the deployment will fail.

C.4 Deployer Properties

The Oracle Virtual Assembly Builder allows you to configure the Deployer properties described in Table C–9 through the deployer.properties file. This file must reside in the `<deployer WLS domain dir>/ovab/config` directory.

Table C–9 Deployer Properties Configurable in deployer.properties

Name	Type	Req'd	Default	Description
phoneHomeTimeout	String	false	900	Configures the phone home request timeout, in seconds.

D

Troubleshooting

The following appendix describes techniques for troubleshooting Oracle Virtual Assembly Builder, and identifies troubleshooting items:

D.1 General Issues

This section describes troubleshooting for general issues.

D.1.1 Error Indicating Another Client is Running

If you see this error when attempting to launch `abctl` or abstudio.sh and you are sure there is no other Oracle Virtual Assembly Builder client running, your machine may have run out of file locks.

You may also see such an error when performing concurrent operations through Oracle Virtual Assembly Builder Studio, or `abctl`. To resolve this problem, check whether there is an environmental issue causing locks to release slowly.

D.1.2 Phone Home Timeouts

To troubleshoot phone home timeouts:

1. Make sure the firewall on your Oracle Virtual Assembly Builder host is turned off.

   As root run the following:

   ```
   $/sbin/service iptables status
   $/sbin/service iptables stop
   ```

   The firewall may be configured to start automatically when your machine reboots. To prevent this you can turn it off completely by running:

   ```
   $/sbin/chkconfig iptables off
   ```

2. The default phone home timeout is 15 minutes. You can increase the timeout in `<Deployer WLS domain root>/ab_instance/config/deployer/deployer.properties`.

   If the file does not exist, create it and add a line similar to `phoneHomeTimeout=<number of seconds>`.

D.1.3 Existing Assembly Archive (OVA file) Prevents Altering of Assembly

If you have built an assembly archive for your assembly then that assembly is locked - meaning you cannot alter its shape, properties, packages or templates until you delete the archive. The archive can be deleted using `abctl delete -archiveOnly -name <assembly name>`.

D.2 Introspection and File Set Capture Failures

This section describes troubleshooting for introspection and file set capture issues.

D.2.1 Introspection Fails with Root Cause 'java.net.ConnectException: Connection timed out'

If you see 'java.net.ConnectException: Connection timed out' as the root cause of an introspection failure you may need to set proxy information in your environment before launching introspection. Enter the settings similar to the following, but appropriate for your shell type:

```
setenv SYSPROPS ' -Dhttp.proxyHost=www-proxy.example.com -Dhttp.proxyPort=80 '
```

After setting this in your environment proceed to launch the Oracle Virtual Assembly Builder Studio graphical user interface, or run your `abctl` command.

D.2.2 Introspection of a VM

Introspection of a deployed virtual machine (VM) can fail to complete when the temporary directory used for introspection is a mounted NFS share with NFS file locking. To avoid this problem, remount the NFS file system with locking turned off:

```
mount -o nolock example:/scratch /net/example/scratch
```

D.2.3 Remote Operation Failures

The logs for remote operations are copied at the end of a remote operation into the local directory of `$AB_INSTANCE/logs/remote_<remote machine name>/`. For example, if the remote machine is abc12345, the logs are stored in `$AB_INSTANCE/logs/remote_abc12345.example.com/`

The default remote working dir is `/tmp/abRemote_<remote username>`, but this can be overridden using the `-remoteWorkingDir` flag.

D.2.3.1 Unable to Connect Errors When Running ipv6 on the Remote Machine

If you are unable to perform remote introspection/packaging and get 'unable to connect' errors, one possibility is that you are running ipv6 on the remote machine but the `sshd_config` file is incorrect.

When remote systems are configured to use ipv6 and ipv4, you must have the following line in the `sshd_config` file:

```
AddressFamily any
```

(and not `AddressFamily inet`).

D.2.3.2 Remote Operation Hangs after Entering Password

If your remote operation hangs after you enter the remote password it may be due to an orphaned remote process left over from a previous remote operation that was killed. If the remote working directory is removed out from underneath the remote process this can happen. Go to the remote machine and kill any orphaned remote processes, then clean up the remote working dir and try your remote operation again.

The remote process appears similar to the following:

```
aime1    11662     1  0 11:51 ?        00:00:01 /tmp/user/abRemote_aime1/ab_
home/jre/jre/bin/java
  -Doracle.core.ojdl.logging.config.file=/tmp/user/abRemote_aime1/ab_
instance/config/logging.xml
  -Djava.util.logging.config.class=oracle.core.ojdl.logging.LoggingConfiguration
  -Djava.security.egd=file:/dev/./your -Dassemblybuilder.spif.app=apps/remotingapp
  -jar /tmp/kaw/abRemote_aime1/ab_home/jlib/oracle.as.assemblybuilder.spif_
0.1.0.jar
```

D.2.3.3 File Permission Problems

Make sure the remote user or sudo user you specify for the remote operation has read permissions for the files in the reference installation.

D.2.3.4 Remote Connection Failure

SSH port forwarding must be enabled on reference systems in order for remote operations (introspection and file set creation) to work properly. Check the ssh config files:

```
~/.ssh/config
/etc/ssh/ssh_config
```

The following error can be encountered if the shell of the remoteUser specified prints things to stdout/stderr during login.

```
Error: Error initializing the remote connection.
Caused by: OAB-90061: Unable to create connection to remote server.
Cause: Timed out trying to connect to IPV4 and IPV6 sockets.
```

Check the profile and rc files for the remote user and take out any logic that does this. Alternatively, you can specify a different remote user and use the sudoUser parameter to specify a user that has permission to examine/capture the reference installation.

D.2.3.5 Remote File Set Capture Failure

The remote working directory must have enough disk space available to store your file sets before they are transferred back to the local machine.

D.3 Template Creation Failures

When a template creation operation fails, check the following:

- Verify you ran `$ORACLE_HOME/oracleRoot.sh` as root during or after installation.
- Verify that the ova utility is installed.

- Verify that you have a valid base image (System.img) and vm.cfg file. Verify that file permissions are correct.
- Verify that you did not run out of disk space.
- Verify that you have a sufficient number of loop devices for the file sets you are capturing. See Section D.3.1, "Insufficient Number of Loop Devices".

D.3.1 Insufficient Number of Loop Devices

You may run into an issue where the number of Linux loop devices on an Oracle Virtual Assembly Builder host are not sufficient to create templates for a generic product with a large number of file sets.

When creating templates, Oracle Virtual Assembly Builder and modifyjeos require one available Linux loop device for each disk in the template (that is, one each for the System.img and AB.img, and one per product disk). A typical Oracle Linux system has only seven loop devices, meaning templates can be created for a template with a maximum of five file sets.

To create templates for an appliance with more file sets, you must create additional loop devices. One way to do this is as follows:

1. Edit /etc/modprobe.conf. Add a line similar to the following:

   ```
   options loop max_loop=<n>
   ```

   Where <*n*> is the number of loop devices you want created.

2. As root, run the following commands to unload and reload the Linux kernel loop module:

   ```
   # /sbin/modprobe -r loop
   # /sbin/modprobe -v loop
   ```

3. Verify that the new loop devices were created:

   ```
   $ ls -l /dev/loop*
   ```

   You should see <n> loop devices.

D.4 Deployer Communication Failures

This section describes troubleshooting of Deployer communication failures.

D.4.1 Invalid Deployer Response Returned

You may run into the following error:

```
Caused by: Invalid deployer response returned.
  Cause: OAB-113409 - An invalid response was returned by the deployer.
  Action: OAB-113409 - Please check the deployer log for additional details.
```

Check the Deployments for the WebLogic Server instance hosting the Deployer and make sure the state of the Deployer application is 'Active' and its health is 'OK'.

D.4.2 401/403 Errors from the Deployer

If you run into 401/403 errors when trying to interact with the Deployer, check the following items in the WebLogic Server Administration Console of the server instance hosting the Deployer.

1. Go to **Deployments** > **Deployer** and verify that the Security Model is "CustomRoles."
2. Go to **Deployments** > **Deployer** > **[Security]** > **[Roles]** > **ApplicationAdmin** and make sure that conditions include Group: CloudAdmins or ApplicationAdmins.
3. Go to **Deployments** > **Deployer** > **[Security]** > **[Roles]** > **CloudAdmin** and ensure that the conditions include Group: CloudAdmins.
4. Go to **Security Realms** > **myrealm** > [Users and Groups] > [Groups] and ensure that "ApplicationAdmins" and "CloudAdmins" exist and are handled by DefaultAuthenticator.
5. Go to **Security Realms** > **myrealm** > [Users and Groups] > [Users] and make sure that "applicationAdmin" and "cloudAdmin" exist and are handled by DefaultAuthenticator.
6. Go to **Deployments** > **Deployer** > [Security] > [URL Patterns] and make sure there is no role definition on `root url-pattern`.
7. When creating a connection in the client, make sure the username is one of the two listed in step 5 and is specified with the password for that user.

D.5 Registration Failures

If your registration has become unresponsive, check the following:

- If you have never had a successful registration, try to register a template using the Oracle VM console directly - bypassing Oracle Virtual Assembly Builder. If this is not successful then the problem is with your Oracle VM environment.
- If you have a very large assembly archive or a slow network you may need to increase the 'Stuck Thread Max Time' setting for both Oracle WebLogic Server where the Deployer is running, and Oracle WebLogic Server running Oracle VM Manager. Access this setting using the WebLogic Server Administration Console, in the *Tuning* tab for the server instance.

D.6 Deployment Failures

This section describes troubleshooting of deployment failures.

If you cannot determine the cause of the failure from the Studio or Deployer logs you'll have to continue investigating. Log in to the Oracle VM Manager console and see if the VMs for your assembly were created and started.

D.6.1 VM Not Created

Check the Deployer and the Oracle VM and Oracle VM Server logs for an indication of why the VMs were not created.

> **Note:** It is possible for the Deployer's state cache to get out of sync from the Oracle VM environment, especially if cleanup type activity was done in the Oracle VM environment outside of Oracle Virtual Assembly Builder. The Deployer may have recorded that an assembly is still registered or deployed, when it has actually been removed from the Oracle VM environment. If this has happened, you must unregister and undeploy your archive through Oracle Virtual Assembly Builder and re-register and redeploy it.

D.6.2 VM Created, But Not Running

Perform the following steps if the VM is created, but not running:

1. Check the Deployer, and Oracle VM and Oracle VM Server logs, for an indication of why the VMs were not started.
2. Try starting the VM manually and see if any useful output is given.
    1. Log in to the Oracle VM Server machine that created the failed VM (you can see which machine in the pool owns the VM via the Oracle VM Manager console)
    2. Find the vm.cfg for the VM in question - it will be in some location under /OVS/Repositories, and will have the ID from the Oracle VM console in its path.
    3. Use the `xm create -f <vm.cfg file>` command to start the VM.
3. Try mounting the disk images and see if any logs were created. (This can be the case if the VM came up but then went back down for some reason). An Oracle Enterprise Linux image is composed of multiple disks; you must mount the disk you are interested in: such as, AB.img, System.img, or Product_001.img.
    1. The Oracle Virtual Assembly Builder logs are on the AB.img disk - but the name of that image changes once it is registered in the Oracle VM environment. To find the location of the newly named image you need to find the vm.cfg. It will be in some location under `/OVS/Repositories`, and have the VM ID from the Oracle VM console in its path.
    2. Figure out the loop device using the path to the image file you just found.

        ```
        #kpartx -a <path to img file>
        #kpartx -l <path to img file>
        ```

    3. From the listing of the previous command, you can determine which loop device has been mapped to the disk. Mount the disk and specify the loop device.

        ```
        #mount /dev/mapper/loop?p? /mnt
        ```

        The first question mark ('?') above represents the number of the loop device, which is usually zero but may be a different number, and the second question mark is the partition number, which is usually zero but may be a different number.
    4. Go to `/mnt` and look at the files. The AB.img may have logs in `/mnt/logs`, if reconfiguration got far enough to create them.
    5. Enter:

        ```
        #umount /dev/mapper/loop?p?
        #kpartx -d AB.img
        ```

D.6.3 VM Created and Running But Cannot be Pinged

The network configuration for the machine did not complete successfully for some reason. If you are using DHCP make sure your Oracle VM environment supports it. If using static IP addresses make sure you have also specified the corresponding hostname in your deployment plan - this is required.

You must specify all of the following network related properties in your deployment plan:

- at the assembly level
  - network_name (needs to match the name of a network in your target (Oracle VM) environment)
- in the network properties for each appliance
  - hostname (if using static IPs)
  - default-gateway
  - dns-domains (only one is supported)
  - dns-servers (only one is supported)
- on each network interface (NIC) for each appliance
  - ip_address (if using static IPs)
  - netmask
  - usedhcp (should be *false* if using static IPs)

D.6.3.1 How to Access a Running VM that Cannot be Pinged

To access the VM:

1. Log in to the Oracle VM Server machine that created the failed VM (you can see which machine in the pool owns the VM via the Oracle VM Manager console.
2. Run the command `xm list`.
3. Find your VM in the list returned - look on the Oracle VM Manager console for the VM ID.
4. Run the command `xm console <vm ID>` and then hit enter and provide credentials (`user`: root, `password`: the password supplied during template creation).

D.6.3.2 Triaging a Network Configuration Failure

To triage the failure:

1. Check the logs under /assemblybuilder/logs. If there are no logs proceed to the next step.
2. Check to see if the ab service was installed. The ab service is installed to `/etc/init.d/ab`. If it is not there, look at the oraclevm-template service log: `/var/log/oraclevm-template`. The oraclevm-template service installs the ab service.

   If the ab service is missing, make sure the permissions on the ab_service.sh and oraclevm-template.sh in your ORACLE_HOME are correct. These files should be executable. If they are not: fix the permissions, recreate your assembly archive, upload, register and try the deployment again.
3. If you believe the late bindings are incorrect or were not sent to the VM you can run the command `/assemblybuilder/etc/vmapi get +`.

   This command will output the late bindings.

D.6.3.3 VM is Created, Started and Can be Pinged

If the VM is up and pingable, then the network configuration for the VM completed successfully.

1. Log in to the failed VM and check the logs under /assemblybuilder/logs. See the Logs section below for details on what is in the various log files. You should be able to ssh to the machine using the root user and the password you specified when creating the templates.
2. If you believe the late bindings are incorrect or were not sent to the VM you can run the command `/assemblybuilder/etc/vmapi get +`.

   This command will output the late bindings.

D.7 Log Locations and Descriptions

This section provides log locations and descriptions. There are several different logs and log locations that are useful to know about when triaging failures.

D.7.1 Studio Logs

The log level for the Studio log can be altered by editing $AB_INSTANCE/config/logging.xml. Change the following line by indicating the desired level:

```
<logger name="oracle.as.assemblybuilder" level="FINE">
```

Local Logs

$AB_INSTANCE/logs/assemblybuilder.log

$AB_INSTANCE/logs/bottler/* - output from the modifyjeos tool used during template creation

Remote Logs

The logs for remote operations are copied at the end of a remote operation into the local directory of `$AB_INSTANCE/logs/remote_<remote machine name>/`. For example, if the remote machine is abc12345, the logs are stored in `$AB_INSTANCE/logs/remote_abc12345.example.com/`

The default remote working dir is `/tmp/abRemote_<remote username>`, but this can be overridden using the `-remoteWorkingDir` flag.

D.7.2 Deployer Logs

The Deployer application log messages will be in the server and/or domain logs for the WebLogic Server instance where the Deployer application is deployed. Stdout/stderr for the WebLogic Server instance may also contain relevant information or stack traces.

<Deployer WLS domain root>/servers/<server targeted by Deployer app>/logs/*

D.7.3 Oracle VM Logs

Oracle VM

/u01/app/oracle/ovm-manager-3/machine1/base_adf domain/servers/AdminServer/logs/AdminServer.log

Oracle VM Server

/var/log/ovs-agent.log

D.7.4 Logs on the VM Instance

/assemblybuilder/logs/ab.out - stdout/stderr and progress messages from Oracle Virtual Assembly Builder infrastructure code and plug-in code

/assemblybuilder/logs/assemblybuilder.log - log messages from Oracle Virtual Assembly Builder infrastructure code and plug-in code

/assemblybuilder/logs/command.out - environment and command details for commands launched via the RehydrateUtils.runCommand() or runCommandAs() methods

/assemblybuilder/logs/proc.<unique>.log - stdout/stderr from processes launched via RehydrateUtils.runCommand() or runCommandAs() where the daemon flag passed in was true

E

Third-Party Licensing

The following appendix contains third-party licensing information. It contains the following sections:

E.1 Java Secure Channel (JSCH) for SSH2

Version: 0.1.44

Vendor: Atsuhiko Yamanaka, JCraft,Inc.

JSch 0.0.* was released under the GNU LGPL license. Later, we have switched over to a BSD-style license.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The names of the authors may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL JCRAFT, INC. OR ANY CONTRIBUTORS TO THIS SOFTWARE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)

HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

E.2 JViews Diagrammer

Version 8.5.

Vendor: ILOG.

None.

E.3 Velocity Engine

Version: 1.6.4

Vendor: The Apache Software Foundation.

This product is licensed under the Apache 2.0 license agreement. See Section E.3.1, "Apache License Version 2.0".

E.3.1 Apache License Version 2.0

The following applies to all products licensed under the Apache 2.0 License:

You may not use the identified files except in compliance with the Apache License, Version 2.0 (the "License.").

You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0. A copy of the license is also reproduced below.

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and limitations under the License.

Apache License

Version 2.0, January 2004

http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other

commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright © 1999-2002 The Apache Software Foundation. All rights reserved.; and (ii) the following statement "This product includes software developed by the Apache Software Foundation (http://www.apache.org/)."

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

E.4 Commons Compress

Version 1.1.

Vendor: The Apache Software Foundation.

E.4.1 Apache License Version 2.0

The following applies to all products licensed under the Apache 2.0 License:

You may not use the identified files except in compliance with the Apache License, Version 2.0 (the "License.").

You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0. A copy of the license is also reproduced below.

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and limitations under the License.

Apache License

Version 2.0, January 2004

http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of,

publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for

reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright © 1999-2002 The Apache Software Foundation. All rights reserved.; and (ii) the following statement "This product includes software developed by the Apache Software Foundation (http://www.apache.org/)."

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

E.5 JSON in Java

Version: None.

Vendor: JSON.org.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

The Software shall be used for Good, not Evil.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.