3

Managing and Monitoring a Composite Application

ORACLE

Objectives

After completing this lesson, you should be able to:

- Deploy and undeploy a composite application
- Test a composite application with Oracle Enterprise Manager and JDeveloper
- Manage SOA composite applications with Oracle Enterprise Manager
- Migrate a composite application to a production environment

Administration of Oracle SOA Suite 12*c* is an extensive topic that merits its own course. In this course, we cover the basic capabilities of the Fusion Middleware Console and Oracle Enterprise Manager web interfaces to administer and manage SOA composite applications.

In many cases, deployment-related tasks can also be performed in JDeveloper and these are discussed where relevant. However, it is unlikely that JDeveloper will be used to deploy directly to a production system.

Therefore, we examine the steps that are needed to create deployment profiles that facilitate managing the life cycle of a composite application and ease the migration from a development to a production environment. The aim of this lesson is to explore the Enterprise Manager web interface, which enables you to initiate a composite application and test it with different data, and learn how to use Enterprise Manager to examine message flow and any fault conditions.

Agenda

- Deploying Composite Applications
- Testing and Managing Composite Applications

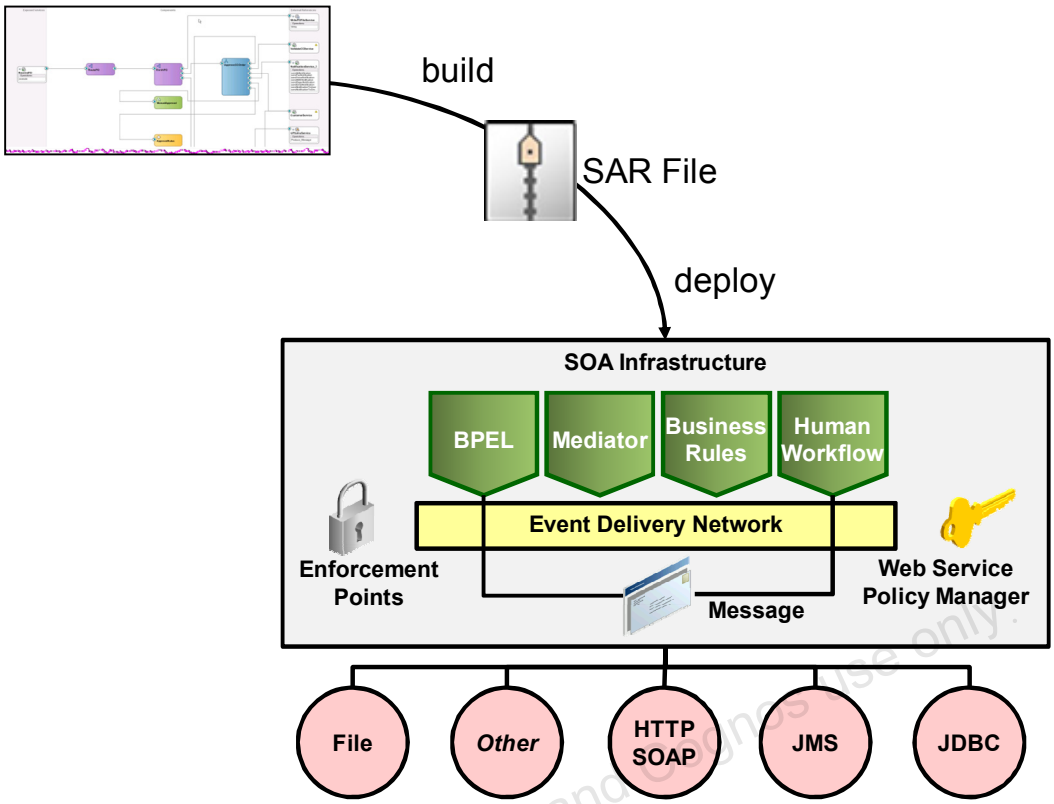
Managing SOA Applications: Overview

Administration of SOA applications involves using the Enterprise Manager web interface for tasks such as:

- Configuring the runtime environment
- Deploying and managing the state of composite applications
- Redeploying or undeploying applications
- Testing composite applications and viewing the results
- Locating composite application instances
- Tracking and monitoring message flow and events in the system

Oracle SOA Suite provides a variety of administrative tools for *monitoring* (observing state or other values in the runtime environment) and *managing (*changing state or runtime parameters) service components and deployed applications. These include tools such as Oracle Enterprise Manager and Fusion Middleware Console. The command-line interface (CLI) tools include WebLogic Scripting Tool (WLST) and Ant. JDeveloper also provides access to certain test, management, and monitoring functionality for the developer.

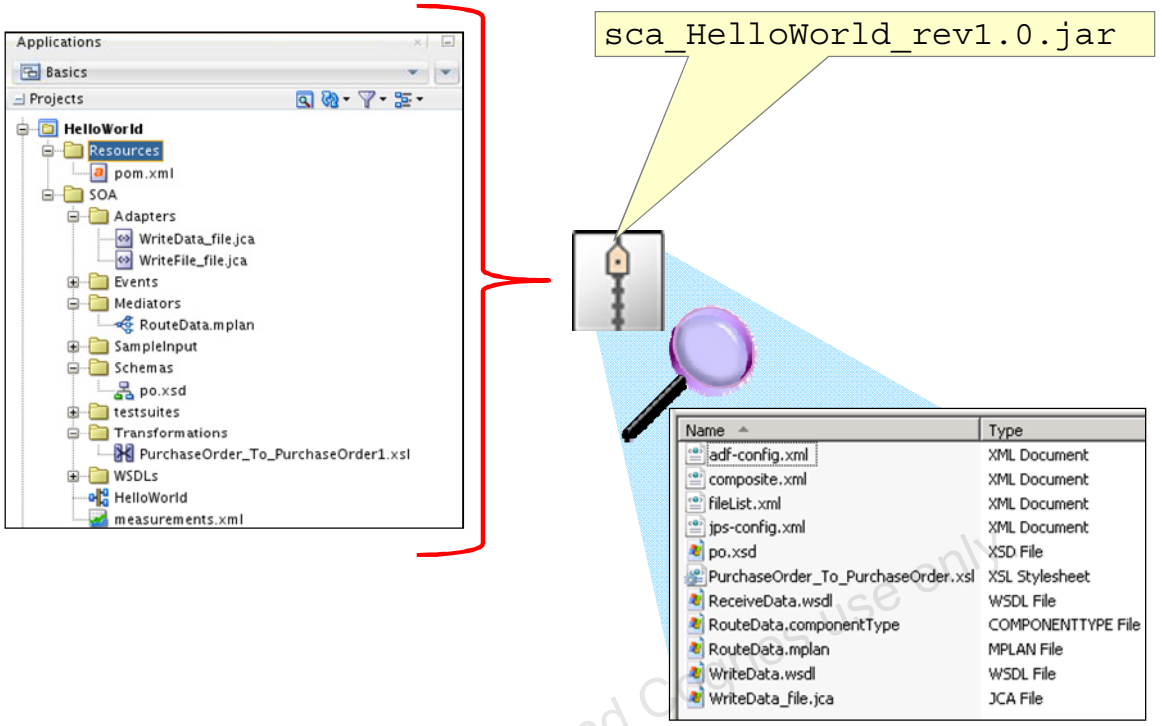
Deploying a Composite Application



When a composite application is assembled and each of its components configured, it is ready for deployment and testing in a runtime environment. Deployment consists of several separate steps:

1. Building the application
   - All service components are included in a file. This file is in the JAR file format, and is known as a Service Archive (SAR) file. The SAR file is a deployment unit that describes and packages service components such as BPEL processes, business rules, human tasks, and mediator routing services into a single application.
2. Deploying the application
   - The SAR file is passed to the target server, which unpacks the archive, and immediately begins executing the application.

Building the Service Archive

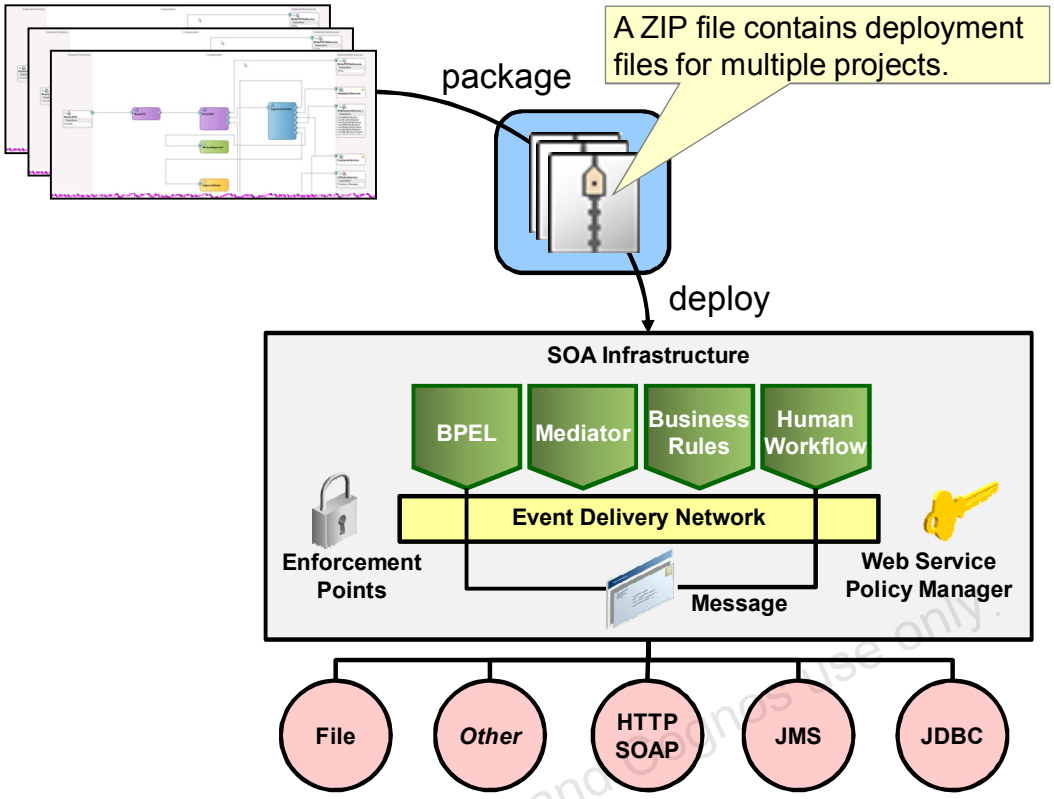


When you deploy a SOA composite application in Oracle JDeveloper, the composite is packaged in a SAR file. SAR is a JAR file (for a single composite application) and is named in the format `sca_<ProjectName>_<revision>.jar`. The SAR file can include the following artifacts:

- Binding components and service components
- References to Oracle Web Service Manager (OWSM) policies and human workflow task flows
- Metadata such as WSDL and XSD files

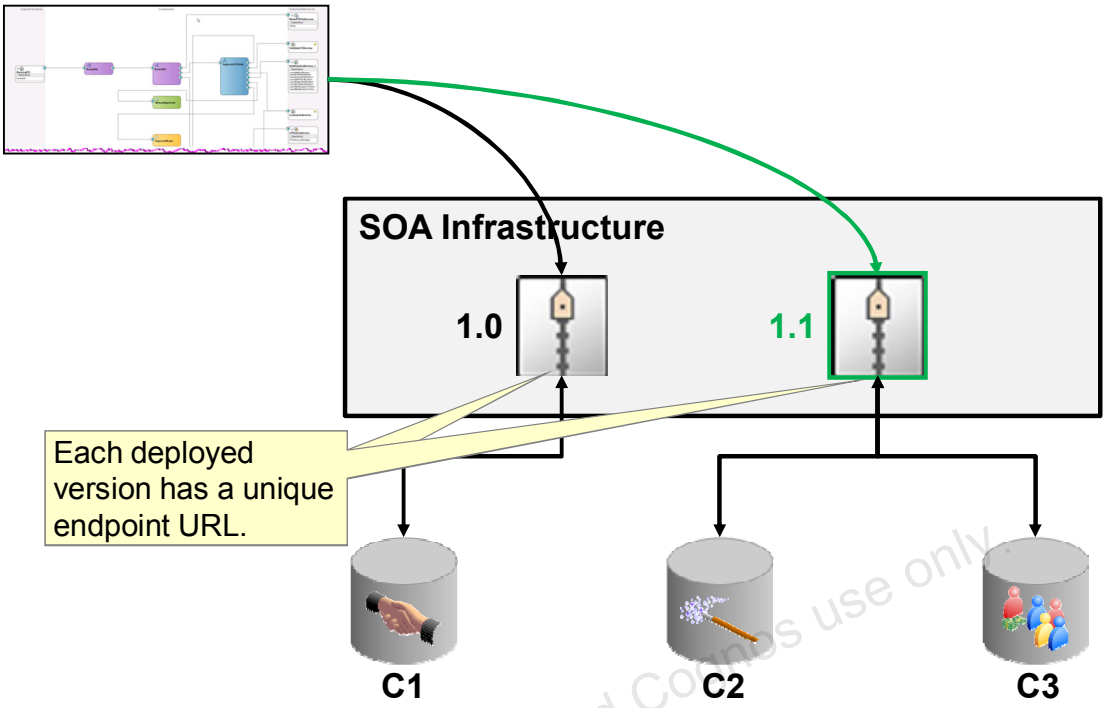
Note: The SAR file is generated in the deployment folder that is specified in the Project Properties settings. By default, this is the `deploy` subfolder of the project.

Deploying Multiple SOA Projects



A JDeveloper Application Workspace may contain multiple projects. Using the Application menu and application properties, JDeveloper enables you to create a SOA bundle to deploy multiple applications with one deploy operation, instead of deploying each separately. Each project should have its own deployment profile—either the default profile or a deployment profile that you create. The SOA Bundler creates a ZIP file that contains one or more SOA archive files. If you deploy using a SOA bundle, all the SOA archives specified on the SOA Bundle Deployment Profile Properties Dependencies page are generated (if they do not exist) and added to the SOA bundle ZIP file. The SOA bundle ZIP file is written to the file name and location specified in the SOA Bundle Deployment Profile Properties General settings. By default, this is a `deploy` subfolder of the Application Workspace directory.

Versioning Applications



A SOA composite application is automatically activated when you deploy it to the SOA infrastructure. During deployment, you can specify a specific *revision number* for the application. A revision is a specific deployed version of the application. You can deploy multiple revisions of an application to support different client requirements at the same time.

The benefit of revisions is that an older version can be executed in the same environment as a new revision. Older client applications are supported until they can be modified to use the new revision. For example, client C1 is using the composite application CA version 1.0 by explicitly specifying the revision number. New partnerships with different clients (C2 and C3) who require the new revision can invoke it either by specifying its revision or excluding the revision information, causing the default revision (the new version) to be invoked. Later, you can plan the migration path for the old client to the newer revision of the application.

The revision value is added to the application name in Oracle Enterprise Manager Fusion Middleware Control Console. Revision 1.0 is the default version for many deployed SOA composite applications. If a new request comes in for a specific composite application revision, that composite application revision is invoked. If a new request comes in without specifying a revision, the default revision is invoked.

Note: An archive selected for deployment can be a SOA composite bundle (containing multiple SOA composite application revisions). However, there should not be any cross-references between composites in the same bundle. For example, Composite A revision 1.0 should not reference Composite B revision 1.0.

Redeploying an Application

Redeploy a composite application if:

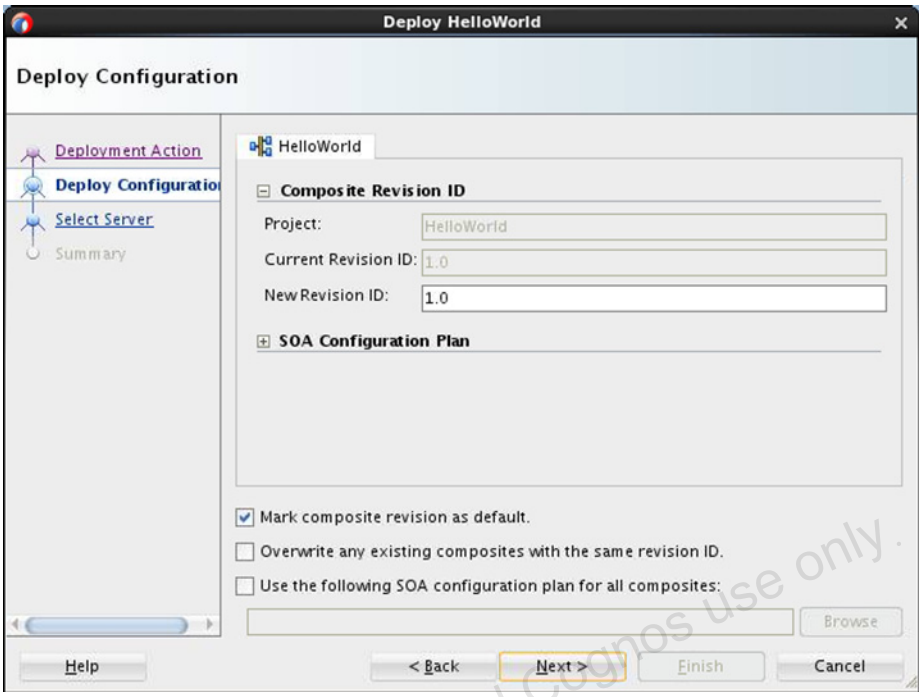
- Configuration and updates to the application do not modify the interface definition
- You wish to make an older version of the application the default version

When you *redeploy* a composite application, you overwrite the specific version with a new copy of the application. If the application is redeployed as the default revision, the active version is immediately replaced, causing any running instances to be terminated. The terminated instances have their runtime state changed to stale.

If you wish to apply configuration and updates to a composite application, and those changes do not modify the interface definition (and therefore, affect clients from a design perspective), the revision ID need not be changed. In this case, redeploying the application makes sense. You might also choose to redeploy to make an older version of the composite application the default revision.

Note: In a production environment, overriding an active version of an application may have negative consequences. Care should be exercised when deciding whether to deploy a new version of an application or when overriding an already deployed version.

Deployment Options



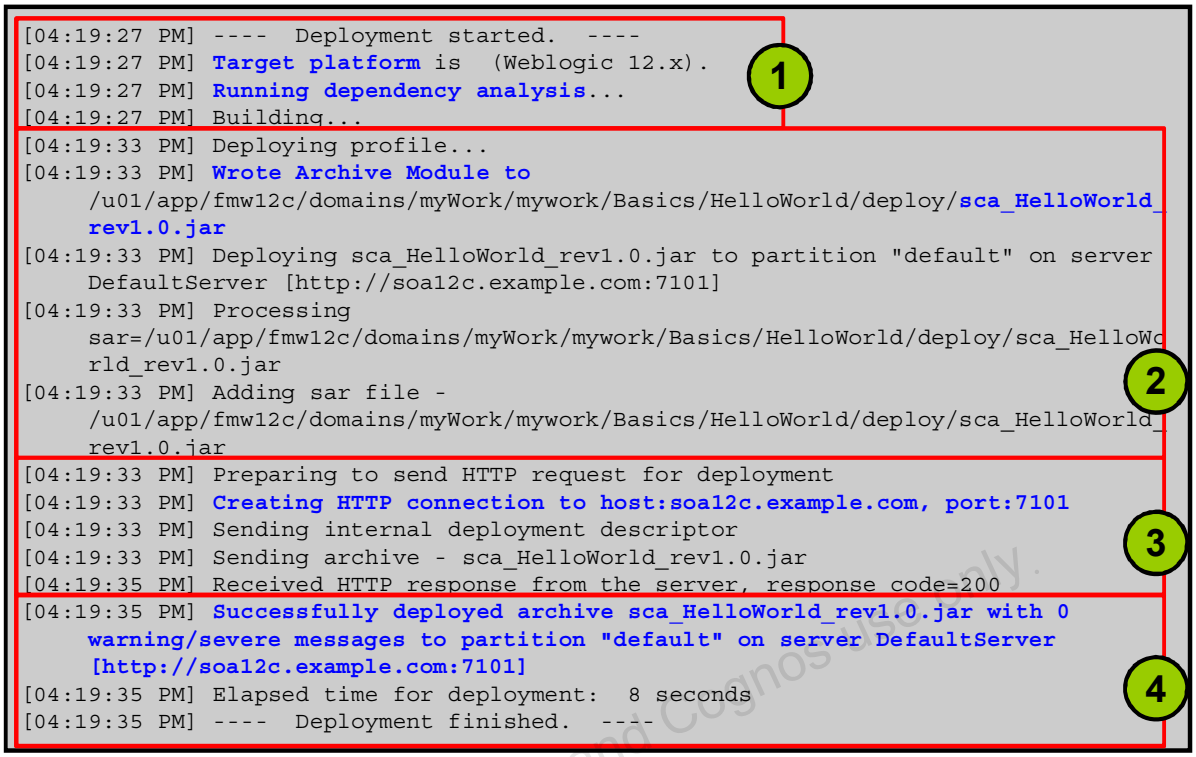
In the SOA Deployment Configuration dialog box, you can manage the following configuration settings:

Project: Displays the project name. However, if you are deploying multiple projects by using a SOA bundle style deployment, the configuration window displays multiple project tabs. The Project fields value changes when you select a project name tab to enable you to view and define the following settings for each project:

- **Current Revision ID:** Is a read-only setting that shows an existing project revision (if previously deployed). By default, this appears as the value 1.0.
- **New Revision ID:** Enables you to change the revision ID of the SOA composite application
  Note: If you do not change the revision ID, you should select the “Overwrite any existing composites with the same revision ID” check box.
- **SOA Configuration Plan section:** Enables you to select a configuration plan file (if any exist) to be applied to the deployed application. When using a SOA bundle to deploy multiple projects, each project may have its own configuration plan attached.

- **Mark composite revision as default:** By default, this check box is selected, making a newly deployed composite revision as the default version invoked for new requests in the runtime environment. However, deselect this check box if you do not want it to be the new default revision.
- **Overwrite any existing composites with the same revision ID:** When this check box is selected, the deployment overwrites an existing SOA composite application with the same revision value. This option should be selected when you do not change revision ID; otherwise, a deployment error occurs.
- **Use the following SOA configuration plan for all composites:** Use this option when you are deploying multiple composite applications with a SOA bundle and when you wish to apply the same configuration plan for all composite applications. Click Browse to locate and select the configuration plan to be used.

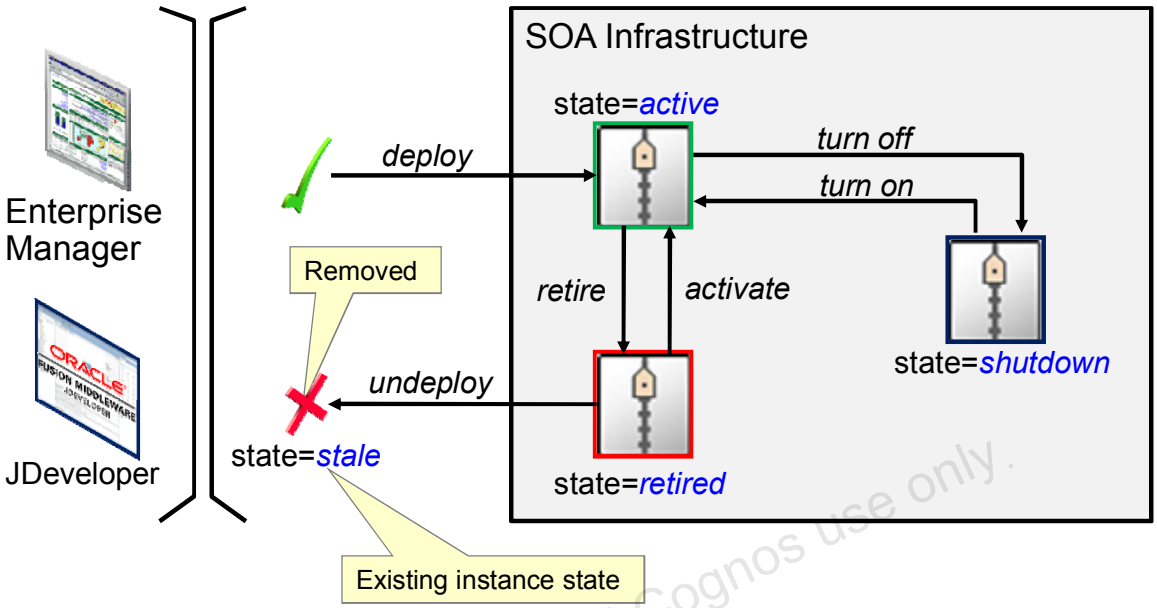
Examining the Deployment Log



The deployment log records each step of the deployment process.

1. When deployment is started, the target platform is identified. The application is checked to see whether it can meet all dependencies for execution on this platform.
2. The project is compiled as a service archive and packaged in a `.jar` file.
3. A connection is made to the target host, and the deployment descriptor is sent with the `.jar` file. The application server deploys the application and returns a response.
4. The response from the application server is displayed in the log file, along with the elapsed time for the entire deployment process.

Application State

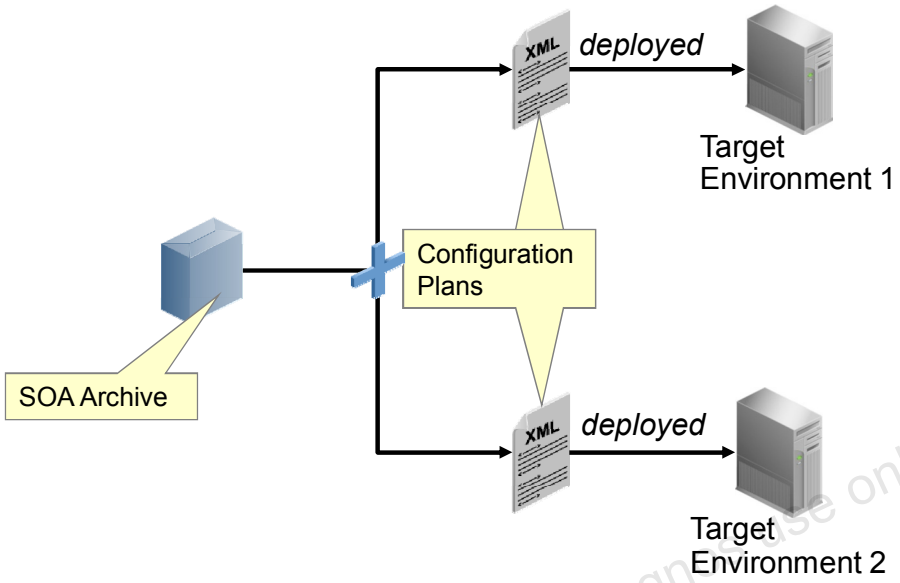


When an application is deployed, it is automatically made *active.* It remains in the active state until some administrative action is taken. Possible administrative actions include turning off or retiring the application. When an application is *turned off* (or "shut down" in Oracle Enterprise Manager), that revision of the application is shut down. New requests to the composite are rejected. An application that has been turned off can also be *turned on* (or "started up" in Oracle Enterprise Manager). When an application is turned on, the application is restarted. New requests are processed. No recovery of messages occurs.

An *active* application can also be retired. When an application is retired, no new instances of that revision of the application are created and any request is rejected. Existing instances are allowed to complete normally. Only active composites can be retired. An application that has been retired can also be *activated*. When an application is activated, new requests are processed and new instances are created.

Undeploying an application removes the selected application revision. You cannot configure, monitor process instances, or view previously completed processes of that revision of the application. The state of any currently running instances is changed to *stale*, and they are effectively terminated. The default revision is changed to the next available revision of the application (if one exists). Applications with running instances should be retired before they are undeployed. This will give all running instances an opportunity to complete.

Configuration Plans: Overview



As you move projects from one environment to another (for example, from testing to production), you typically must modify several environment-specific values, such as Java Database Connectivity (JDBC) connection strings, host names of various servers, and so on. Configuration plans enable you to modify these values by using a single text (XML) file called a *configuration plan*. The configuration plan is created either in Oracle JDeveloper or with command-line tools such as the `ant-sca-text.xml` Ant script. During process deployment, the configuration plan searches the SOA project for values that must be replaced to adapt the project to the next target environment.

Thus, you can create multiple configuration plans for different target environments without directly modifying the original source code that was created during development.

Modifying a Configuration Plan

```
<SOAConfigPlan ...>
  <composite name="EchoMediatorComposite">
   <import>
     <searchReplace>
     <search>soa12c_dev.example.com</search>
     <replace>soa12c.example.com</replace>
     </searchReplace>
   </import> ...
  </composite>
</SOAConfigPlan>
```

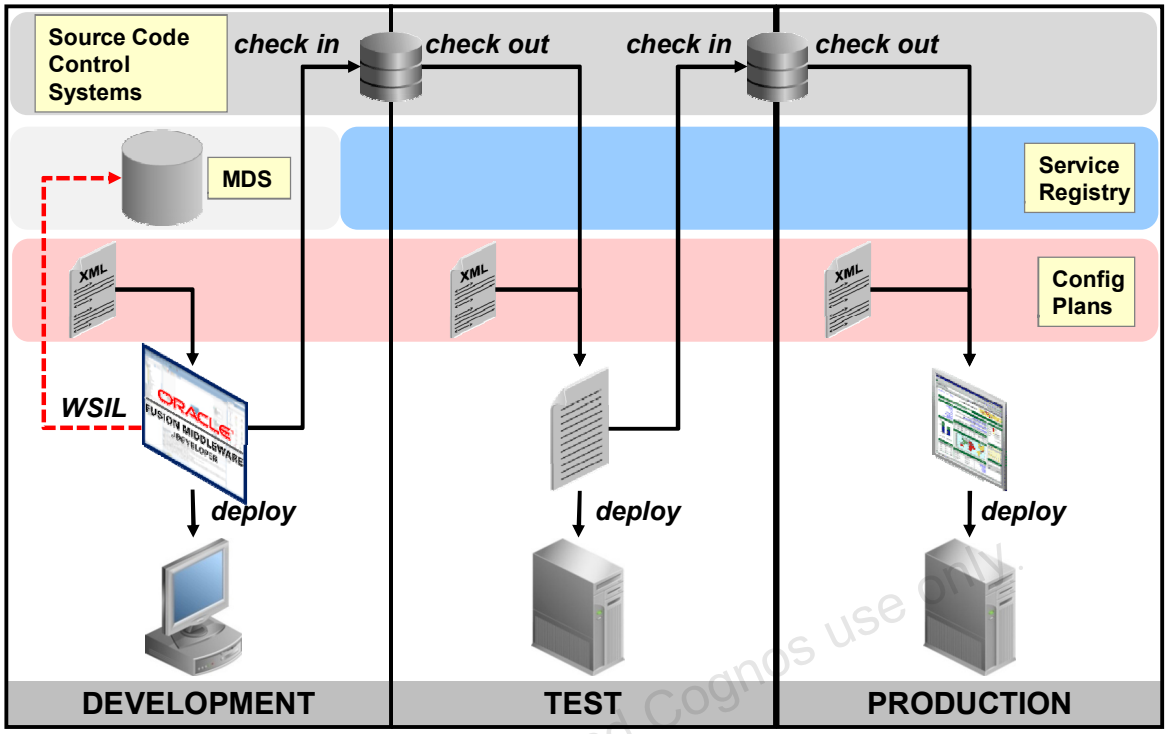

Modifying the configuration plan is relatively straightforward, by including the appropriate <searchReplace> elements and their <search> and <replace> child elements with appropriate values. The slide example shows how to provide search and replace host name strings that are found in the schemaLocation attribute of a WSDL import element.

You can modify the configuration plan file to replace the following attributes and properties:

- Any composite, service component, reference, service, and binding properties in the SOA composite application file (composite.xml)
- Attribute values for bindings (for example, the location for binding.ws)
- The schemaLocation attribute of an import in a WSDL file
- The location attribute of an include in a WSDL file
- The schemaLocation attribute of an include, import, and redefine in an XSD file
- Any properties in JCA adapter files
- Modify and add policy references for service and reference binding components

Note: Editing the configuration plan requires that you know the assembly model structure and all the service artifacts used by the composite application. For more information, refer to the Fusion Middleware documentation titled *Developing SOA Applications with Oracle SOA Suite.*

Managing the Life Cycle of a Composite Application



The slide illustrates one possible scenario for migrating an application from development to test and production environments. Each phase of migration can use any of these tools:

- Oracle JDeveloper 12*c* is typically used during the development phase. With JDeveloper, you use a source code control system (SCCS) for version control of the composite application source code and its configuration plan.
- Ant is a command-line tool that can be used to deploy a composite application with the deployment plan modified for the test environment. After successful testing, the composite application and test deployment plan are checked in to the SCCS.
- Oracle Enterprise Manager 12*c* Fusion Middleware Control can deploy the composite application with the deployment plan modified for the production environment.

The deployment configuration plan simplifies the process of migrating a composite application across environments.

Note: Service lifecycle management ideally includes an enterprise repository and a service registry for central storage, access, and versioning of services, subject to SOA governance requirements. Without an enterprise repository or a service registry, you can use the Oracle SOA Suite 12*c* Metadata Services (MDS) Repository to store reusable artifacts, which can be accessed through JDeveloper during the development phase.

Quiz

After an application is deployed, its state must be set to active before it is executed by the application server.

a. True
b. False

Answer: b

False. When an application is deployed, it is automatically made *active.* It remains in the active state until some administrative action is taken.

Agenda

- Deploying Composite Applications
- Testing and Managing Composite Applications

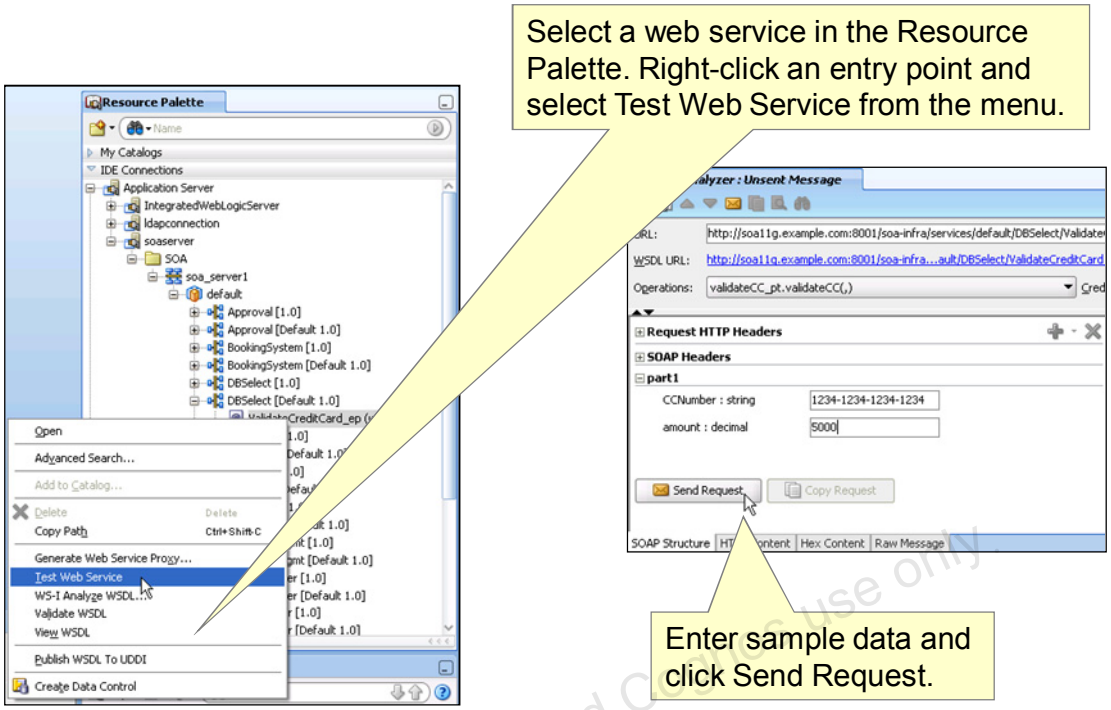
Testing a Composite Application

Composite applications can be tested by using:

- Oracle Enterprise Manager Fusion Middleware Control, which enables:
  - Initiating composite application instances
  - Viewing responses for synchronous requests
  - Tracking message flow through a composite application
  - Viewing faults and log files
  - Managing fault conditions
- Oracle JDeveloper Web Service Test tool, which enables:
  - Initiating composite applications
  - Viewing responses for synchronous requests
  - Examining the HTTP request and response headers

Both Oracle Enterprise Manager Fusion Middleware Control and Oracle JDeveloper have testing tools that enable you to initiate composite applications and view responses (for synchronous services). The slide highlights some of the capabilities of each tool, and highlights some of the differences. Each has its place in the set of tools that you can use to test your applications or the services that they reference.

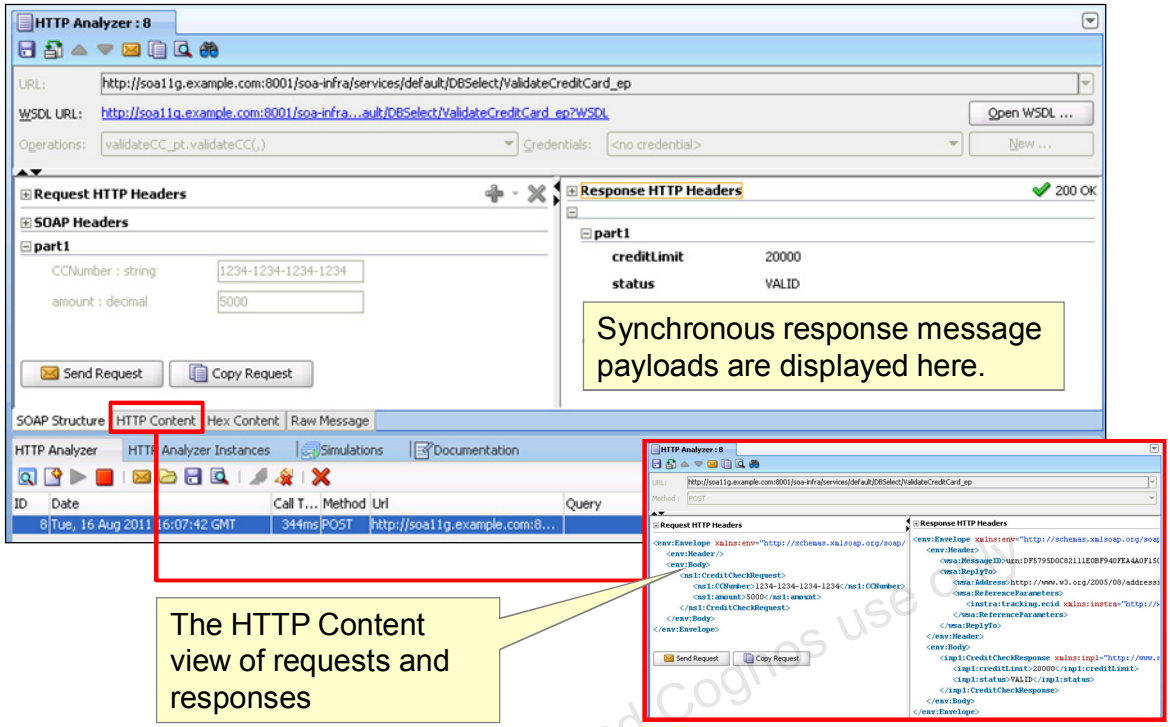
Initiating a Test in JDeveloper



The JDeveloper web service testing tool is called HTTP Analyzer. It is a proxy server that is capable of intercepting HTTP communications as well as creating, copying, and modifying SOAP packets to send to services and interpret service responses.

To initiate a test of a web service in JDeveloper, select the desired service in the Resource Palette. Right-click the service and select Test Web Service. The request and response panes are displayed. Enter sample data and click Send Request.

Viewing Responses in JDeveloper



If the service that you are testing is a synchronous service, the response or fault message is displayed in the reply window area.

If you click the Raw Message tab, you can view the HTTP header lines and the SOAP content in XML format. The HTTP Analyzer tab in the lower section of the window enables you to view request messages for all exchanges. Just click a numbered line to view the information exchanged for that request-response cycle. You can also initiate additional requests.

Managing SOA Applications with JDeveloper

The Application Server Navigator in JDeveloper enables you to perform the following actions on SOA applications:

- Deploy a SOA archive.
- Retire an existing revision.
- Turn an application OFF and ON.
- Set the default revision.
- Undeploy the application.

To open the Application Server Navigator, select JDeveloper View > Application Server Navigator. Expand the Application Servers tree and the target server connection. Perform the desired action from different context menus.

Right-click the SOA folder to view its context menu. Select Deploy SOA Archive to deploy a SOA archive (SAR) file that has already been created.

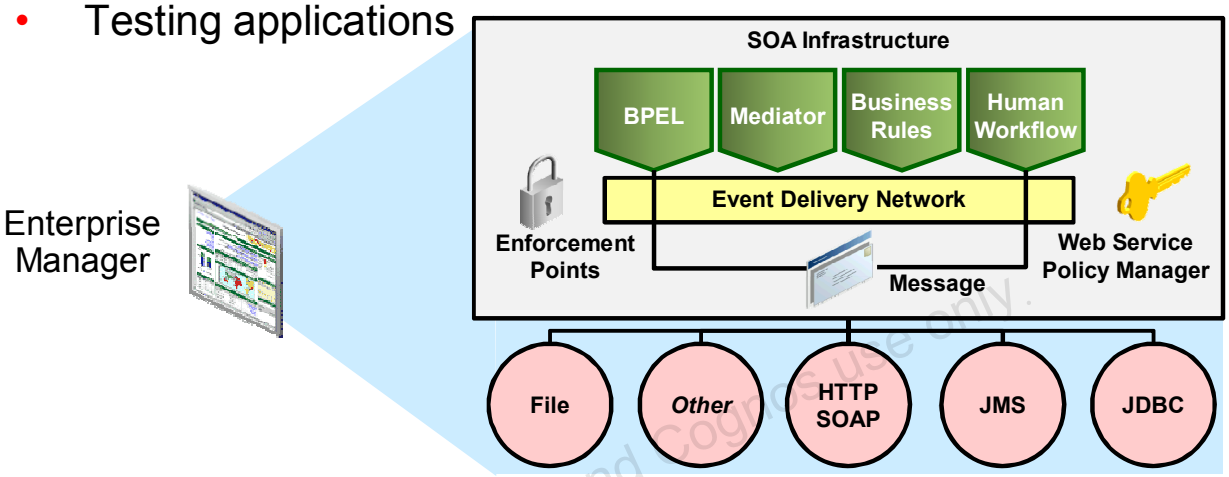
Right-click any composite application node and select one of the following operations:

- **Retire** to stop new instances from being created before you undeploy an application
- **Turn OFF** to shut down the application so as to prevent new instances from being created. This is similar to performing the Shutdown action in the Oracle Enterprise Manager Fusion Middleware Control. When you turn off an application, the context menu item is changed to “Turn ON” so that you can enable the application to start instances for receiving new requests from clients.
- **Set Default Revision** to alter the version of a composite application that is used to handle requests. This menu option is not active (disabled) unless you have multiple revisions deployed.
- **Undeploy** to remove the composite application from the server

Enterprise Manager

Enterprise Manager provides tools for many purposes, including:

- Deploying and undeploying applications
- Managing the state of applications
- Testing applications



Oracle Enterprise Manager 12*c* Fusion Middleware Control is the primary tool for managing, monitoring, and configuring the SOA runtime environment and components.

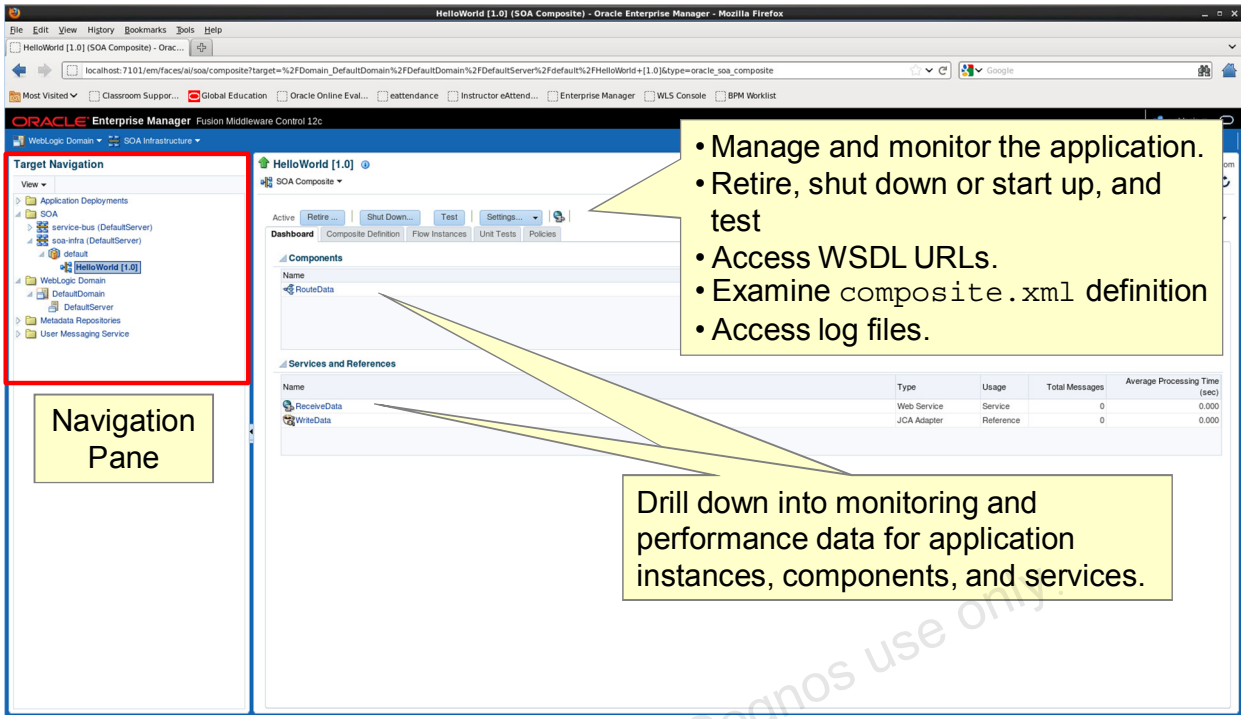
By using Fusion Middleware Control, you can perform tasks such as:

- Deploying and undeploying composite applications
- Managing the runtime state of composite applications
- Shutting down and restarting the composite applications
- Initiating tests of SOA applications by using a web-based service testing tool
- Tracking and monitoring composite application instances and message flows through a composite application
- Examining and managing application fault conditions
- Monitoring component engines

From Fusion Middleware Control, you can navigate to Oracle SOA Suite administration tasks through the SOA Infrastructure home page and menu. The SOA infrastructure provides you with access to all deployed SOA composite applications, service engines, service components, business events, and other elements.

Fusion Middleware Control provides a wide variety of administrative and performance data for the SOA components, composite applications, and composite instances within the SOA infrastructure, thus enabling you to administer and pinpoint issues.

Oracle Enterprise Manager: Overview

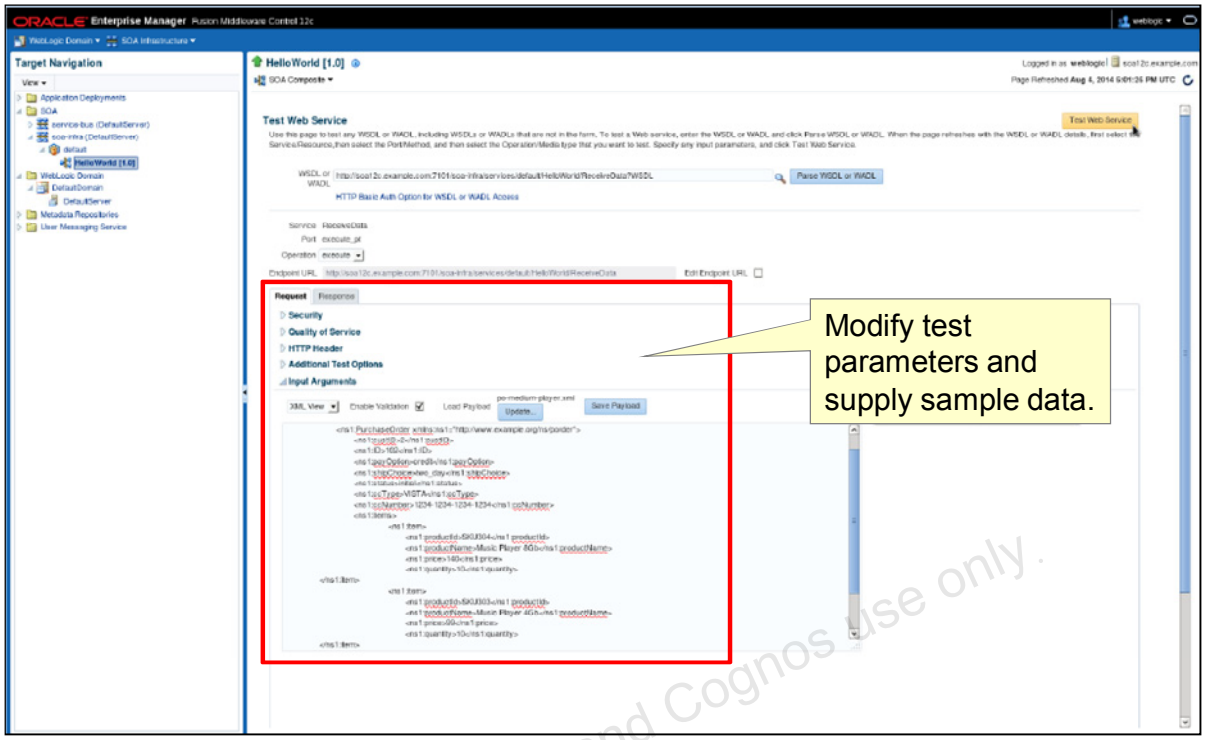


The home page for a given application includes buttons and tabs that provide access to a variety of application management functions. The buttons allow you to:

- Manage the state of an application (retire and shutdown or startup)
- Test the application
- Access WSDL URLs
- Examine the `composite.xml` definition
- Access log files

By default, the dashboard tab is displayed. From there, you can drill down into monitoring and performance data for application instances, components, and services. Other tabs allow you to search for specific instances and faults and rejected messages. You can also access unit test cases and security policies from there.

Testing a Composite Application



To initiate a test of a composite application, navigate to the home page for the composite application that you wish to test, and click Test. The Test Web Service page is displayed (shown in the slide).

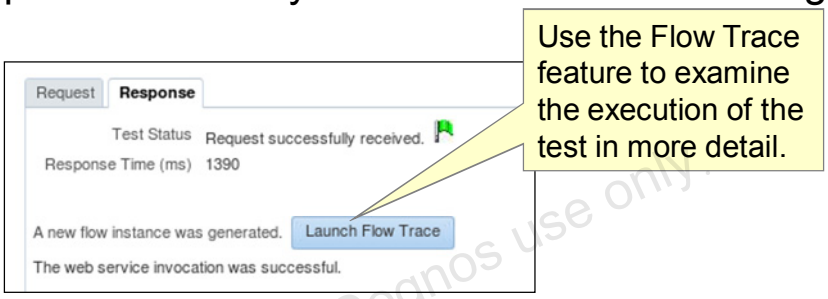
On the Test Web Service page, click Test (if there is one entry point) or click Test and select one of the entries from the Test drop-down list (if multiple entry points are available) Select an operation from the Operation drop-down list (which may provide a list if the service has more than one operation).

On the Test Web Service page, scroll down the page and enter various properties in the Request tab section. In particular, enter request data values in the fields supplied in the Input Arguments section, which by default displays input fields (based on the request messages structure) in Tree View mode. Click Test Web Service to invoke the service.

Response Tab

The webpage is refreshed with the Response tab displaying either of the following:

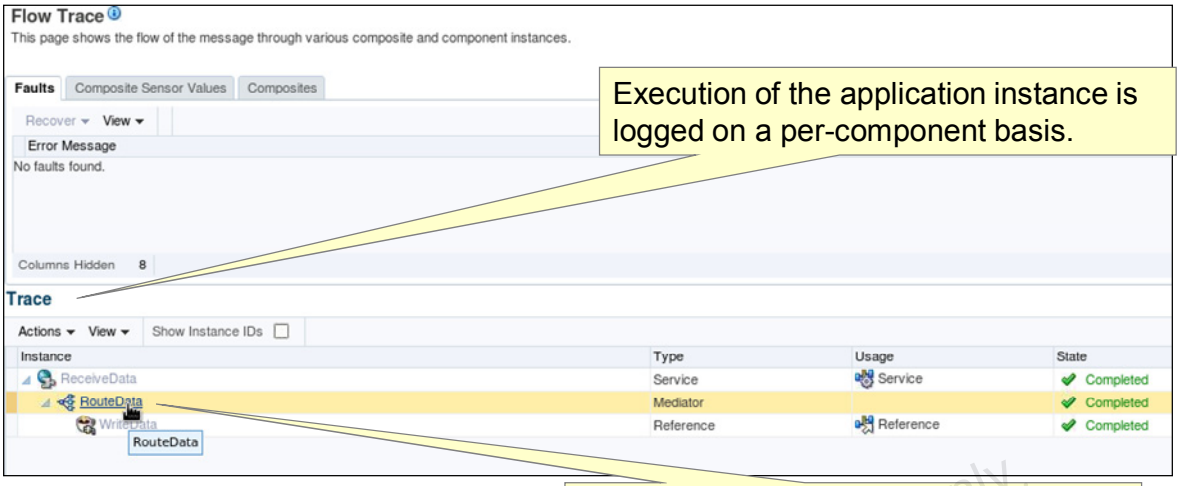
- The synchronous response data or fault message
- The fact that no response was possible because it was an asynchronous request (regardless of whether it was a one-way or two-way operation), because the web client cannot receive a response as an asynchronous callback message



When the test is run, the Response tab is enabled. If the service is synchronous, the tab displays the response data or fault message. If the request is asynchronous, the window displays a message that the test is complete but no response is possible.

From the Response tab, the Launch Flow Trace button provides access to step-by-step status of the test.

Flow Trace



The Flow Trace logs the execution of an application instance on a per-component basis. The component's type, current state, and time stamp of the last update is noted in the trace. Further drilldown is possible by clicking a component name.

Quiz

The Flow Trace logs the execution of an application instance on a per-component basis.

a. True
b. False

Answer: a

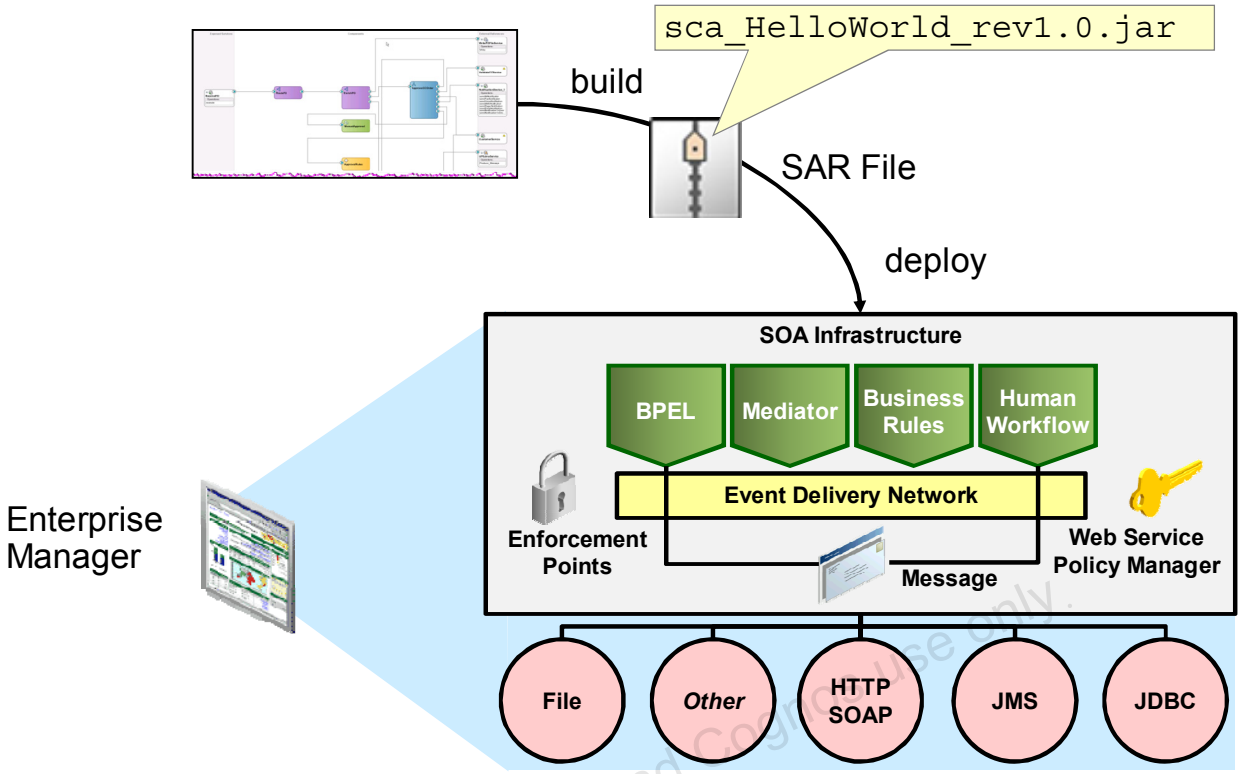
True. The Flow Trace logs the execution of an application instance on a per-component basis. The component's type, current state, and time stamp of the last update is noted in the trace. Further drilldown is possible by clicking a component name.

Summary

In this lesson, you should have learned how to:

- Manage SOA composite applications by using Oracle Enterprise Manager
- Track messages through SOA composite applications by using Oracle Enterprise Manager
- Deploy a composite application
- Undeploy a composite application
- Migrate a composite application to a production environment

Practice 3-1 Overview



In this practice, you create, deploy, and run a simple SOA composite application. You use the Enterprise Manager web application to test the service with sample input. The goal of this practice is to become familiar with the development environment, and to begin forming an understanding of composite application components.

Practice 3-2 Overview

In this practice, you modify the HelloWorld File adapter to use a logical name. Your tasks are to:

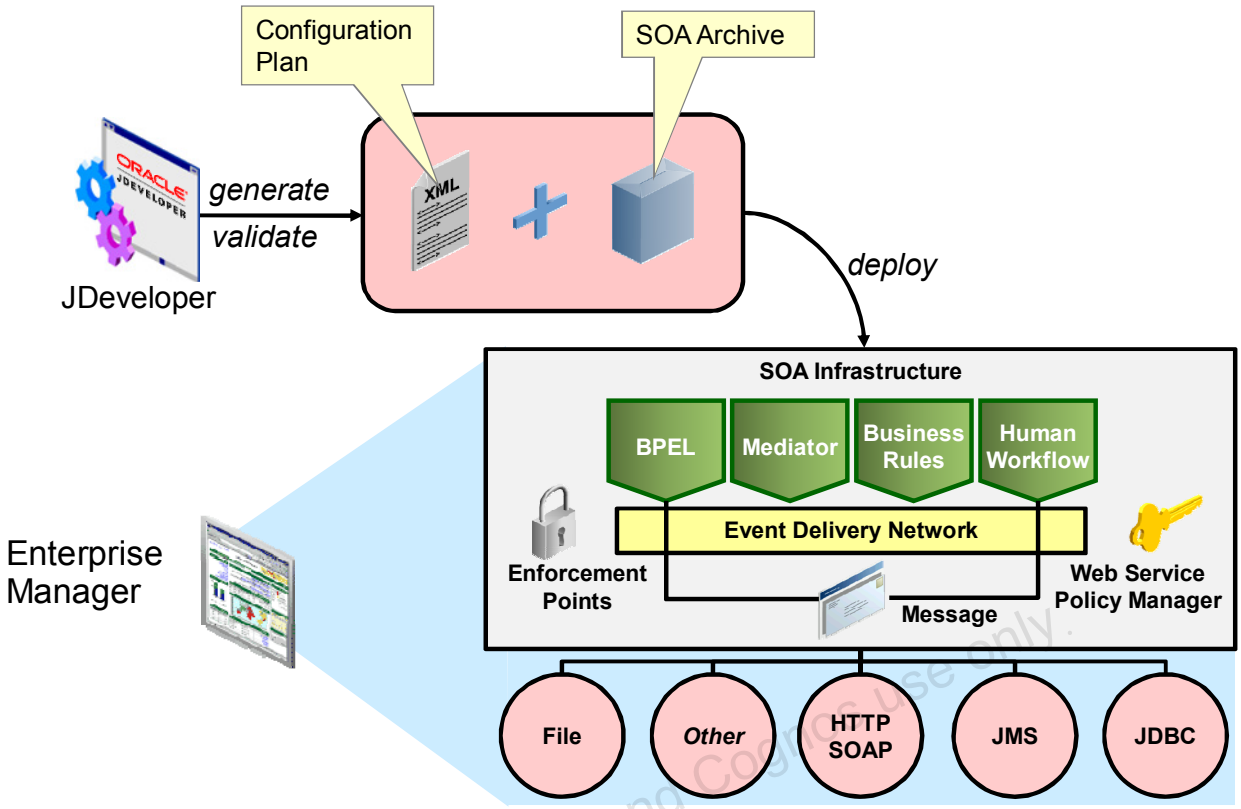
- Modify the File adapter definition of the HelloWorld application to use a logical name instead of a physical directory name for the location to write files in
- Set a reference property of the File adapter to define the logical name value
- Deploy the updated HelloWorld application and test that the order file is saved in the directory that is specified for the logical name

ORACLE

In Practice 3-2, you perform the following:

- Modify the File adapter definition of the HelloWorld application to use the logical name `orderfiles` instead of specifying the physical directory name for the location to write files in.
- Set a reference property of the File adapter to define a value for the `orderfiles` logical name.
- Deploy the updated HelloWorld application and test that the order file is saved to the directory specified for the logical name.

Practice 3-3 and 3-4 Overview



In the final two practices for this lesson, you perform the following:

- Generate a configuration plan for the HelloWorld project that redefines the value assigned to the logical name property of the File adapter.
- Validate the configuration plan.
- Redeploy the HelloWorld composite application with its new configuration plan by using Enterprise Manager.