# Oracle® SDP Provisioning

Implementation Guide

Release 11i

August 2000

Part No. A86196-01

ORACLE®

Oracle SDP Provisioning, Realease11i

Part No. A86196-01

# Contents

# Send Us Your Comments

**Oracle SDP Provisioning, Release 11i**

**Part No. A86196-01**

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, please indicate the document title and part number, and the chapter, section, and page number (if available). You can send comments to us via the postal service.

Oracle Corporation
CRM Content Development Manager
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

If you would like a reply, please give your name, address, telephone number, and (optionally) electronic mail address.

---

---

---

If you have problems with the software, please contact your local Oracle Support Services.

# Preface

Welcome to the Oracle SDP Provisioning, Release 11i.

This Detailed Implementation Guide provides information and instructions about the implementation of the Oracle SDP Provisioning application.

This preface explains implementation considerations and processes is organized and introduces other sources of information that can help you.

## Intended Audience

This guide is aimed at the following users:

- Product Implementation team (Oracle and Customer)
- Oracle and Customer Project Managers
- Technical Support Associates
- System Administrators (SAs), Database Administrators (DBAs), and others with similar responsibility.

This guide assumes you have the following prerequisites:

- Understanding of the product implementation processes.
- Knowledge of Oracle Network Logistics operation and services
- Basic understanding of Oracle and Developer / 2000
- Understanding of the interface protocol to each of the fulfillment elements (telnet, script)
- Background in SQL, PL / SQL, SQL* Plus programming

## Structure

This manual contains the following chapters:

- Considerations for planning
- User Roles
- Typical Release Dependencies
- Setting up Oracle Provisioning
- Workflows
- Testing an Implementation Project
- Considerations for Future Upgrades
- Appendices
  - Public APIs
  - SDP Parameters
  - Configuring Adapters

# Related Documents

For more information, see the following resources:

## URLs

- http://crm.us.oracle.com
- http://sdp1-nt.us.oracl.com/sdp/sdphome.html
- http://nplab-pc.us.oracle.com
- http://products.us.oracle.com

## Published Resources

- *Oracle Provisioning User's Guide*
- *Oracle Number Portability User's Guide*
- *Service Delivery Platform, Developer's Guide*
- *Developing Number Portability Applications Reference Guide*
- *Oracle Provisioning Technical Reference Manual*
- *Developing XML Message Based Application Reference Guide*
- *Con figuration "How To's"*
  - User Defined Workflows
  - Workflow Fulfillment Actions
  - Timers
- *Aim Documentation*

# Conventions

The following conventions are also used in this manual:

| Convention | Meaning |
|---|---|
| .<br>.<br>. | Vertical ellipsis points in an example mean that information not directly related to the example has been omitted. |
| . . . | Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted |
| **boldface text** | Boldface type in text indicates a term defined in the text, the glossary, or in both locations. |
| < > | Angle brackets enclose user-supplied names. |
| [ ] | Brackets enclose optional clauses from which you can choose one or none. |

# Implementing Oracle Provisioning

## Considerations for Planning an Implementation Project

The following items are part of planning an Oracle Provisioning Implementation:

- Overview of Oracle Provisioning
- Application architecture
- Features and functions
- User roles
- Other considerations

### Provisioning Defined

One of the primary functions of the Oracle Service Delivery Platform is to supply telecommunication providers, ISPs (Internet Service Providers) and similar vendor companies with the ability to manage the critical business functions of order capture and order activation, network administration and operation.

#### The Benefits of Oracle Provisioning

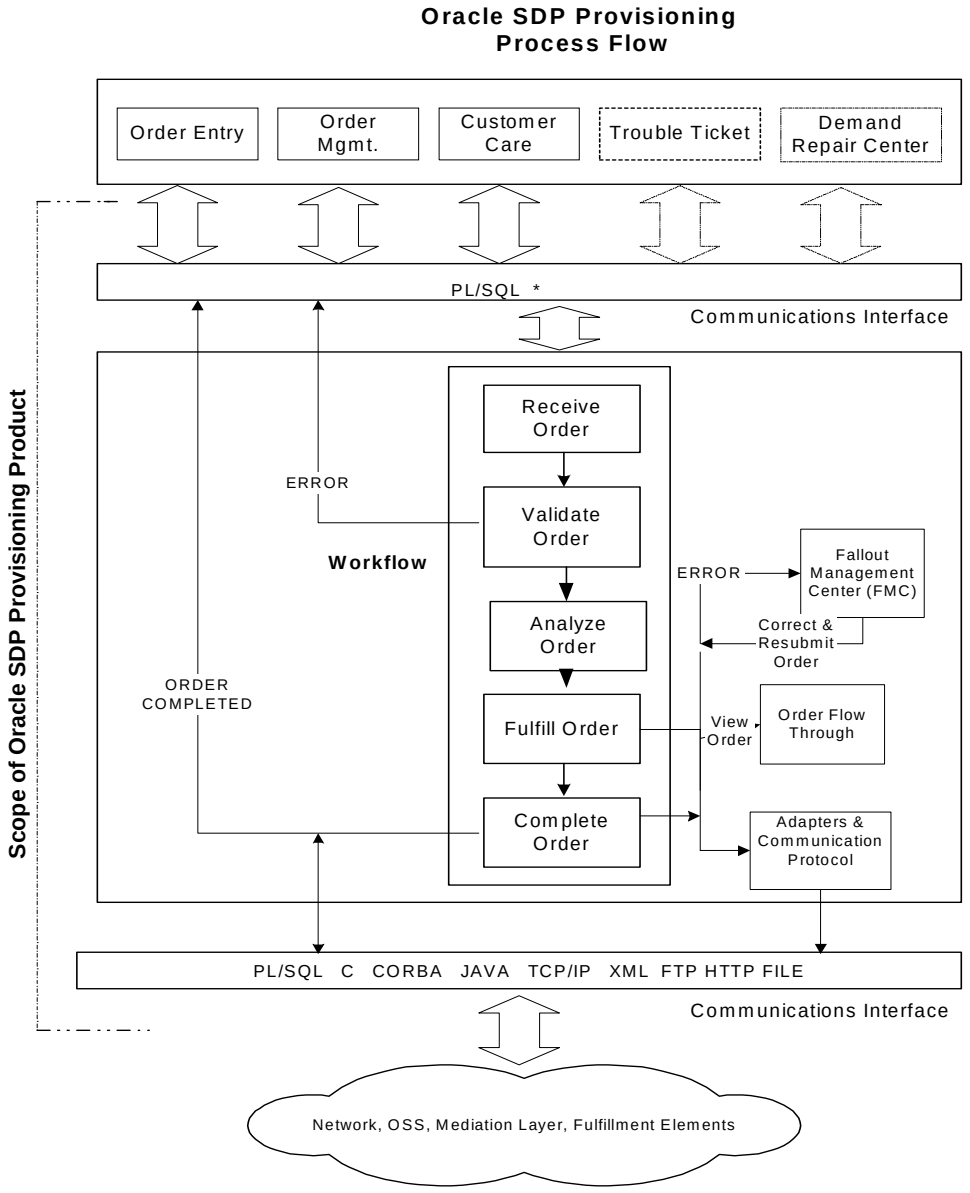
Oracle Provisioning provides the following benefits:

- A fully automated, nearly real-time activation of services for multiple network elements in a multivendor network environment.
- A number of APIs (Application Program Interfaces) that you can use as building blocks in creating an interface to any Operational Support System (OSS) available on the service provider's network. For example, you can use these APIs to create an interface to an external order entry or customer care system.

### Features of Oracle Provisioning

Oracle Provisioning is designed to be an integrated management system that acts as the coordinating link between different external systems. It supports the following features:

- The rapid deployment of new services by integrating the functions of service management with the network details such as service attributes, provisioning algorithms, and fulfillment elements.
- The provisioning of multiple technologies such as ATM, ADSL, switches, routers.
- The delivery of complex services requiring configuration of multiple fulfillment elements and OSS (Operational Support Systems) interfaces.
- The delivery of bundled services combining voice, data and video products.




* SDP provides wrappers that can be written to support other platform languages (other than PL/SQL).

## Application Architecture

### Oracle Provisioning Architecture

The Oracle Provisioning architecture consists of three functional layers:

- **The application interface laye**r

  The application interface layer performs the mediation function with external systems. This layer is made up oaf API sets implemented through PL/SQL remote procedure calls.

- The workflow management layer

  The workflow management layer performs order processing.

- The communications management layer

  The Communications Management layer is responsible for managing communications between SDP and the network elements. This activity is handled though the Connection Management Utility (CMU) module.

### Components

The basic components which make up Oracle Provisioning are as follows:

- Order Capture through Process_Order API (see Appendix A: Public APIs) and Order Entry simulation interface.
- Ability to Resubmit Orders
- Order Management through the Flow Through Module
- Connection Management Utility acting as a gateway to OSSs, FEs, NEs
- Order Fallout and Repair through the Fallout Management Center
- Service Definition through the Service Creation Manager Module
- Order Queue Processing Management through the System Console
- Set-up of Service Providers
- Ability to Create Messages for send/receive to any external system.

## Features and Functions

Oracle Provisioning provides three major functions:

### Provisioning

Through PL/SQL procedures calls, Oracle Provisioning provides mechanisms for performing the following tasks:

- Submitting orders
- Obtaining order statuses
- Receiving error notification
- Responding to error notifications

The Provisioning function also provides the capabilities for Service Creation, Package Creation, Workitem Creation, Workflow Customization, handling of Fulfillment Elements, handling for Fulfillment Element Types, and other Provisioning Procedure capabilities. These capabilities will be covered in greater detail during the Instructor Lead Training (ILT).

### Administration

Oracle Provisioning has established procedures for User Administration, Security, Console, New Product Configuration, New Service Configuration, and Connection Management (through the Connection Management Utility - CMU), througput of order processing (through the System Console). These subject will be covered in greater detail during the ILT.

### Operations

Oracle Provisioning Operations consists of Flow Through Management and Fallout Management (through the Fallout Management Center (FMC). These two modules are discussed in the following section and are covered in even greater detail during the ILT.

## Service Order Requests in Oracle Provisioning

Oracle Provisioning handles a typical service order request by performing the following tasks:

1. Capturing a service order request
2. Validating the order
3. Analyzing the order
4. Fulfilling the order
5. Completing the order
6. Managing order fallout (if needed)

### Capturing a Service Order Request

Initially, a customer places an order with the carrier for a telecommunication service. Examples of services are phone line connections, Internet connections or wireless access.

Customer orders are captured from an external order entry system. If necessary, you can use the Oracle Provisioning screens to input customer orders. However, as the name of the form indicates the purpose of the Test Center. The primary capture process is through the Oracle Provisioning APIs.

### Validating the Order

After the service order is captured by the system (or manually entered, as the case may be), it is examined for validity by the system. If an order fails validation, OP returns an appropriate message indicating such errors to the order originator. Failed orders due to validation errors are not processed further in Oracle Provisioning.

Order validation includes verifying:

- The order syntax
- The inclusion of mandatory parameters
- The inclusion of necessary dependent information
- The supplied dates are valid

### Analyzing the Order

After the system captures an order, Oracle Provisioning performs the following steps:

1. It first analyzes the order.
2. It then breaks down the order into the required set of tasks needed to complete the order.
3. Finally, it orders the tasks so that they will be performed in the proper sequence.

For example, say a new order requires two new lines to be activated at a customer residence. The first line is an analog voice phone line. The second line is a digital ISDN line to be used for high-speed Internet access. In addition, each separate line is to have a different bundle of services associated with it. The analog line must have basic voice service, as well as the Caller ID and Call Waiting features. The digital line needs to have a connection established to the desired ISP (Internet Service Provider).

After order capture, Oracle Provisioning analyzes and validates the order, and then puts the order steps into sequence. While some of the tasks involved in setting the customer order can be performed in parallel, others must be performed in a definite sequence. In this example, you must establish basic voice service on the analog line before implementing the additional features.

### Fulfilling the Order

Oracle Provisioning accomplishes order fulfillment through one of two ways:

- OP performs the requested tasks internally.
- OP directs external systems to perform the tasks involved in setting up the order.

Oracle Provisioning first translates these tasks into network language, and then activates the various network elements needed to complete the tasks.

For example:

- First, it can request configuration and activation of services on specific switches. (Connections to the needed network elements are managed using the Connection Manager.)
- Then, it can notify the OSS (Operational Support Systems) to commence these service.

Workflow manages the tasks involved in setting up an order. Workflow is either internal to Oracle Provisioning or external to it.

- **Internal workflow:** This is the default, or standard workflow delivered with the application.
- **External workflow:** You create this workflow, and integrate it with the standard workflow delivered with the application. For example, you write the business process logic (the executable code) to retrieve data from outside the Service Delivery Platform database. (This could be, for example, the telephone number function for a network switch.)

You configure workflow to match your company specific business practices.

> **Note:** Oracle Provisioning can also handle the results of the network elements through its generic fulfillment engine.

### Completing the Order

A provisioning order can complete either successfully, or unsuccessfully.

- **Order Completion Successful:** The system sends a notification message of success to the Event Manager that includes both the internal and external order IDs for the order, as well as details of the services it provisioned. These details includes all the subscriber details, the service assignments, parameter values, and the product attributes. The system captures (stores) all activation information, updates and modifications performed on the service order request.

  The Event Manager then publishes the event that the service order request completed successfully. The system changes the order status to COMPLETED.

- **Order Completion Unsuccessful:** Oracle Provisioning suspends the provisioning process after the first error occurs. It then creates a notification message with the error information and sends the message to the Notifications utility. The order remains open, but the system changes the order status to SUSPENDED. Order processing can not continue until you rectify the error.

  To continue processing, you must modify the order via the Notifications utility interface and resubmit the order to the provisioning process where the error occurred.

### Managing Order Fallout

Order fulfillment can fail during the provisioning process due to invalid activation information, or perhaps the unavailability of a needed network element.

Oracle Provisioning provides a number of ways to effectively manage the order fallout process through its Notifications utility.

You can, for example:

- Examine order details and drill down to error messages
- Track the progress of problem resolutions through the system
- Initiate action to reconfigure a service, if needed
- Generate a notification and forward the notification to the Trouble Ticket system and / or Work Force Managements
- Confirm that the order trouble has been cleared

### Why Provisioning Orders Fail

Order fulfillment can fail during the provisioning process due to the following factors:

- **Invalid orders:** Orders that fail the order validation process may require escalation to the order entry or customer service groups. In such cases, the Notifications utility serves as a tracking and notification mechanism.
- **Bad or missing data:** Incorrect or missing data can cause order fallout.
- **Fulfillment action problems:** Fulfillment elements may not function properly, be too busy, or be unavailable, so the system rejects the service activation command.
- **Time-outs on orders:** The connection session between the application and the fulfillment elements may time-out due to either application or to external errors. Orders that have timed out may have been partially completed prior to the time-out.
- **Network element or service configuration problems:** The new service order may conflict with the current subscriber configuration. This may be the result of invalid orders. However, there may be other reasons, such as improper configuration of the customer.

Systems, such as Queues, will fail if concurrent managers are not running.

### Ways to Resolve Failed Orders

You use the Notifications utility to track, monitor and resolve failed orders.

The table following details how you can resolve failed orders.

| Action | Description |
|---|---|
| Retry Failed Orders | You can re-send failed orders in one of the following ways:<br>▪ Modify parameters in the failed order and re-send the order down to where the failure occurred, and then continue processing.<br>▪ Re-send an unchanged order down to where the failure occurred in the provisioning process. (Failure may have been due merely to network problems.)<br>▪ Restart the provisioning workflow from the point where it failed or stopped. |
| Stop Processing Failed Orders | You can stop the workflow processing of an order that is due for provisioning. Stopping the workflow is useful in the case of identified network outages and order processing backlog. |
| Terminate Failed Workflows | You can completely terminate processing an order if the order encounters an exception state. |
| Resolve Failed Workflows | You can correct the error in the order provisioning workflow (from within the Notifications utility). |
| Escalate Failed Workflows | You can escalate a failed order notification by forwarding it to another user or group to troubleshoot the order. |
| Re-assign Failed Workflows | You can reassign a failed order to another user or group for resolution. |

### System Notification Types

As an order moves through the provisioning process, the system can generate notifications in response to various events.

Example events are:

- Recovery deamons
- Service provisioning process failure
- Internal exceptions

### Notification types

The workflow produces two types of notifications:

- **Informative notifications:** The system generates information notifications if the Service Delivery Platform system detects a stopped fulfillment element manager process and starts it up automatically. Also, the system generates this kind of notification if an internal problem occurs that requires a system administrator's attention.
- **Order fallout notifications:** The system generates an order fallout notification for failed orders.

### The Order Resubmission Utility

You use the Resubmission Utility to force the system to re-run an order on a fulfillment element. However, only those orders which have either completed, or have been previously submitted to the fulfillment element in question, can be resubmitted. You resubmit an order, for example, in the event that a network switch fails or experiences a service disruption, causing a provisioning order to fail.

Through the Resubmission Utility, you specify a particular fulfillment element and the time window during which it was unavailable. After entering the required information, the fulfillment element re-executes the fulfillment actions as scheduled. For each resubmitted order, the system generates a unique Job ID. You can use the Job ID to track the progress of the resubmitted order.

### Resubmitted Orders and Fulfillment Actions

Successful re-execution of the fulfillment action requires that the adapters run in this mode.

- If no adapters are available, and the fulfillment element adapter maximum has **not** been reached, then an error message is generated.
- If no adapters can be started, all re-submission requests are queued.

You use the Resubmitted check box in the Order Flowthrough utility to identify resubmitted fulfillment actions.

### Resubmitted Orders and Notifications

Resubmitted orders execute the same fulfillment procedure as original orders.

- These resubmitted orders can generate notifications based on the fulfillment procedure logic.
- Any generated notification message includes the Job ID.

## The Event Manager

The Event Manager handles incoming and outgoing messages within the Oracle Service Delivery Platform. The Event Manager provides a set of APIs that internal and external systems use to register for messages.

- Incoming messages are frequently a response to a previously initiated message from an external or internal system
- Outgoing messages are frequently user-defined notifications informing an external or internal system of an event occurrence.

### Message Registration

External applications can send request messages to the Event Manager, if they are registered for that message. External applications can also register for response messages.

### Inbound Messages

The Event Manager identifies the subscriber and delivers an incoming message to the registered applications. The following sequence of steps is performed automatically. The Service Delivery Platform:

- Picks up the message from the Incoming Queue
- Pulls the message out of the Incoming Queue and into the Event Manager
- Validates the message structure via the Event Manager
- Verifies if any internal or external subscribers are registered for this message
- Executes the user-defined logic
- Delivers the message to all registered application systems

### Processing Logic

If no application has registered for a message, then the application uses the default processing logic after message validation. External systems, however, can define the processing logic at the time when validation logic is defined.

## Queues in Oracle Provisioning

As a service order moves between the different stages in its processing, the order s often queued in one of the many internal system queues.

As the provisioning of an order progresses, the order moves sequentially through the various system queues as part of the fulfillment process. For example:

- Incoming orders are moved to the Orders Pending queue if scheduled for later provisioning.
- Orders are transferred from the Orders Pending queue to the Orders Ready queue as the provisioning date occurs.

The table following describes the queues that the Oracle Provisioning system uses.

| Queue | Contains |
|---|---|
| Orders Pending | Orders waiting for a provisioning date to occur |
| Orders Ready | Orders that are ready to be provisioned immediately |
| Work Item | Work items for orders currently being processed<br>(Work items are a logical grouping of fulfillment actions.) |
| Fulfillment Actions | Fulfillment actions that are ready for execution<br>These can be either:<br>- Conditional fulfillment actions<br>- Fulfillment procedures<br>- User-defined workflows<br>(Fulfillment actions execute fulfillment procedures on fulfillment elements.) |
| Wait for FE Channel | Fulfillment actions waiting for one of the channels configured for the fulfillment elements to becomes available |
| FE Ready | Fulfillment actions associated with each fulfillment element that are ready to be executed |
| Outbound Messages | Messages sent by Oracle Provisioning |
| Inbound Messages | Messages received by Oracle Provisioning |

| Queue | Contains |
|---|---|
| Event Manager | Events which occur internally in Oracle Provisioning<br>Example events:<br>▫ FA_done<br>▫ Work Item_done<br>▫ Lineitem_done |

## The Oracle Provisioning System Queues

As an order moves through the system, it progresses from queue to queue in a sequential manner. Typically, this progression moves through the system queues in the following order.

- Order Pending Queue
- Order Ready Queue
- Work Item Queue
- FA Queue

### Order Pending Queue

The Process Order API moves the order into the Order Pending queue. The order stays in the pending queue until the date is set for provisioning. This date can be passed to the Service Delivery Platform as part of the order.

On the provisioning date the order is moved to the Order Ready queue. If an order does not contain a provisioning date, the default date becomes the current date and the order is moved immediately to the Order Ready queue.

### Order Ready Queue

The Order Ready queue holds orders that are ready to be provisioned immediately. The order is dequeued from the Order Ready queue and the work items generated by the work-plan are placed on the work item queue.

### Work Item Queue

The system uses the work item queue to manage the execution of work items.

Processing the work item involves the following actions:

- Checking the configurator for existence of fulfillment actions mapped to the work items
- Determining if the work item is a regular Service Delivery Platform work item or a user-defined workflow

The following then occurs:

- With regular work items, the fulfillment actions are enqueued into the FA Queue in the sequence specified.
- For user-defined workflows, the workflow determines the execution of the fulfillment actions. (Users can drag and drop EXECUTE_FA activities into their workflows based on the business process being implemented.)

> **Note:** At this point in the provisioning process, the Service Delivery Platform does not know which fulfillment elements need to be provisioned.

### FA Queue

The system uses the FA Queue to manage the execution of fulfillment actions for regular work items.

Processing the fulfillment actions involves:

- Checking the configuration to determine which fulfillment element is to receive the fulfillment procedure
- Checking the availability of an adapter for the fulfillment element

The following then occurs:

- If an adapter is available, then it is immediately used to send the fulfillment element commands specified in the fulfillment procedure.
- If all adapters for the fulfillment element are busy, or none are immediately available, the system enqueues the fulfillment action in the Wait for FE Channel queue.

## Other Queues

The following queues are used as necessary:

- Outbound, Inbound and Event Queues
- FE Ready Queue
- Wait for FE Channel Queue

### Wait for FE Channel Queue

When an adapter becomes available, the system moves the fulfillment action waiting in the queue to the FE Ready queue.

### FE Ready Queue

This is a temporary holder for all fulfillment actions that are waiting for a free fulfillment element adapter. As an adapter becomes available, the fulfillment action is immediately dequeued and its fulfillment procedure executed.

### Outbound, Inbound and Event Queues

These queues are used by the Service Delivery Platform to communicate with asynchronous XML message based systems. They communicate using the File and TCP/IP Java adapters.

## The Queue Console

The Queue Console provides a set of screens that display specialized information about each internal system queue. It can also be used to track down an order and to take corrective actions to expedite the order fulfillment process.

The Queue Console provides the following information about each queue:

- State Information
- Process Information
- Entry Information

### State Information

The Queue Console displays information on the state or status of each Oracle Provisioning queue.

- **Enabled:** The processors are up and running and processing orders.
- **Disabled:** The processors are running, but an error condition has occurred. No order is being processed.
- **Suspended:** The processes have been suspended manually.
- **Shutdown:** The processors have been shutdown manually.

### Process Information

The Queue Console displays information on the processes that are registered or running for a queue.

- **Processors Registered:** The number of processors pulling or processing orders out of a queue.
- **Processors Running:** The number of orders running in a queue.

### Entry Information

The Queue Console displays information the number and date of the entries in a queue.

- **No. of Entries:** The number of entries in the queue.
- **Last Entry Date:** The date when the last order entered this queue.

## Network Adapters

A network adapter mediates the communication between Oracle Provisioning and the fulfillment elements.

- An adapter is a runtime instance of a fulfillment element.
- Messages to be sent and received are specified by the fulfillment element and the connection procedures executed by Oracle Provisioning workflow.
- The provisioning workflow executes connect procedures for each new fulfillment element connection (provided a procedure has been defined for the particular fulfillment element).
- The Connection Manager interacts with fulfillment element connections through adapters.

### Adapter Usage

The following table lists the ways that the Service Delivery Platform uses adapters, and describes each:

| Usage | Description |
|---|---|
| Provisioning Mode | ▪ Usual mode of operation<br>▪ Required mode for executing a fulfillment action or fulfillment procedure.<br>▪ Only one fulfillment action or fulfillment procedure can be processed through an adapter execution.<br>▪ An adapter using this mode can service only one request at a time. |
| Messaging Mode | ▪ The available message types are File and FTP (File Transfer Protocol).<br>▪ These adapters send and receive messages.<br>▪ An adapter using this mode can service multiple requests from multiple sources at any given time. |
| Test Mode | ▪ Used for synchronous provisioning of services.<br>▪ Used for the Direct Request Center order types.<br>▪ A fulfillment action can use only one adapter at a time. |
| Order Resubmission | ▪ Orders for a certain fulfillment element are resubmitted using the Order Resubmission Utility.<br>▪ When processing is done, adapters run in this mode. |

### The Network Adapter Types

Network adapters mediate communications between the Oracle Service Delivery Platform and the fulfillment elements in the system.

- Oracle Workflow executes connect, disconnect, and provisioning procedures for each new fulfillment element connection. (That is, provided procedures are defined for those particular fulfillment elements.)
- The Connection Manager interacts with the fulfillment element connections through adapters.
- Oracle Workflow executes any fulfillment and connection procedures specified.

The table following lists the types of adapters that are used in the Service Delivery Platform, and describes each:

| Adapter Types | Description |
|---|---|
| Interactive | ▪ Simulates a Telnet, FTP, or any other type of interactive session.<br>▪ Requires a connection procedure. |
| Script | ▪ Executes items, such as UNIX command line scripts, shell scripts, or similar executables. The scripts can be of any type, including JavaScript.<br>▪ Does not require a connection procedure. |
| File | ▪ Used to build an XML file and to upload the file via FTP client to a remote fulfillment element type.<br>▪ Can be used to process batch file commands, if the fulfillment element requires this procedure |
| HTTP | ▪ Communicates directly to a web application server such as WAP server using the HTTP protocol. The Oracle Service Delivery Platform writes an HTTP client (a robot) that communicates with the fulfillment element type.<br>▪ Can be used whenever the fulfillment element supports an HTTP protocol. |

### Adapter Commands

You can issue commands to network adapters in one of the following ways.

- You pre-define a sequence of commands (through the Connection Manager) for the system to perform automatically during the course of the provisioning process. This method is the most common way of issuing commands to the adapters.
- You issue adapter commands manually, also through the Connection Manager. However, while possible, this is not recommended.
- You schedule commands to run at a future time, or times.

You request, or schedule a request, that an adapter perform one of the following operations:

- Suspend
- Resume
- Disconnect
- Connect
- Shut Down
- Startup

The following table identifies the available commands that the provisioning system sends to an adapter, the state that the adapter must be in to accept that command, and the state of the adapter after the command is implemented.

| Operation | Current State | Next State |
|---|---|---|
| Suspend | Idle | Suspended |
| Resume | Suspended | Idle, Busy |
| Disconnect | Idle | Disconnected |
| Connect | Disconnected | Idle, Busy, Error |
| Shutdown | Busy, Idle, Suspended, Disconnect, Session Lost | Shutdown |
| Startup | Shutdown | Idle, Busy, Error |

### Suspend

This command stops the application from sending orders to be provisioned. It places service orders in a queue until the fulfillment element becomes available.

If this command is used, then the following events occur:

- The application will not send any data / commands / Service Order Requests to the connected fulfillment element.
- All service order requests that specify this adapter are stored in a job queue until the adapter is re-started.

  The current status of the adapter must be *Idle* for this command to be available.

### Resume

This command acts as a toggle to end / resume normal operations from a previous suspension.

If this command is used, then the following events occur:

- The Connection Manager sets any fulfillment element interfacing to this adapter to either *Idle* or *Busy*.
  - It sets the fulfillment element to **Idle**, if the job queue is empty for the adapter interfacing at the fulfillment element (meaning that no service order requests awaiting provisioning use this adapter).
  - It set the fulfillment element to **Busy**, if there is a service order request in the provisioning queue that uses the adapter interfacing to the fulfillment element.
- The application continues the provisioning process, if service order requests are waiting in the job queue. The next service order request marked for processing continues its provisioning course.

The current status of the adapter must be *Suspended* for this command to be available.

### Disconnect

This command is used to close the appropriate connection. It stores the service activation order in a job queue until the connection to the fulfillment element is restored by issuing a connect request.

If this command is used, then the following events occur:

- The Connection Manager sets the adapter status to Disconnected.
- The Connection Manager places all service order requests that use this adapter into a queue until the adapter is re-connected.

The current status of the adapter must be *Suspended* or *Idle* for this command to be available.

### Connect

This command is used to establish a connection to a fulfillment element so that is starts, or continues, its normal operation. Service order requests that are stored in a job queue re-start processing.

If this command is used, then the following events occur:

- The Connection Manager sets the fulfillment element to one of the following states:
  - Idle
  - Busy
  - Error
- Processing continues based on the following conditions:
  - If service order requests are waiting in the job queue, the next service order request due for processing continues its provisioning course. The status of the adapter is *Busy*.
  - If the job queue is empty for that fulfillment element (meaning no service order requests are awaiting provisioning), the status of the adapter is set to **Idle**.
  - If the Network Element Manager status is *Error*, then the connection could not be established to the Network Element, the connection procedure failed.

The current status of the adapter must be **Disconnected** for this command to be available.

### Shut Down

This command is used to close the connection to a fulfillment element by discontinuing adapter operation. Service order requests are stored in a job queue until a new adapter becomes available.

If this command is used, then the following events occur:

- The Connection Manager sets the adapter status to *Shutdown*.
- The Connection Manager shuts down the adapter.

The current status of the adapter can **not** be *Shutdown* or *Reconnecting* for this command to be available. If the status is *Busy*, the shut down operation does not happen immediately. The application logs a *Shutdown* request that becomes active upon order completion.

### Start Up

This command is used to start up a new adapter and establish a connection to a fulfillment element.

If this command is used, then the following events occur:

- The Connection Manager sets the fulfillment element status to **Idle** or **Busy**.
- Order service requests start processing using the new adapter.
  - If service order requests are waiting in the job queue, then the next service order request continues its provisioning course. The status of the adapter is *Busy*.
  - If the job queue is empty for that fulfillment element, then the status of the adapter is set to **Idle**.

The current status of the adapter must be **Shutdown** for this command to be available.

## Adapter Modes

There are two types of adapters modes. They are:

- Startup Mode
- Debug Mode

### Startup Mode

There are three possible states to the Startup Mode:

- Automatic

  The adapters can be started from the command line using the **xdp-ctl** utility.

- Manual

  The adapters can be started only through the Connection Manager.

- Disabled

  Adapter operations are disabled unless the adapter mode is changed to either *Automatic* or *Manual*.

### Debug Mode

This mode indicates the level of logging for the UNIX executable and the interactive adapter types.

- Low

  Critical activities only are logged.

- Medium

  Most of the system activities are logged.

- High

  All activities are logged.

- None

  No activities are logged except **fatal** errors.

## Network Adapter Management

You issue requests (commands) to the adapters and schedule adapter activities through the Connection Manager.

### Adapter Commands

You issue the following commands to adapters:

- **Connect:** Establish a connection to the fulfillment element and resume its normal operation.
- **Disconnect:** Close the appropriate connection and store the service activation order in a job queue until the connection to the fulfillment element is restored by issuing a connect request.
- Ftp file: Transfer a file using ftp.
- **Resume:** Resume normal operation.
- **Shut down:** Close the connection to the fulfillment element by bringing down the adapter.
- **Start up:** Start up a new adapter and establish a connection to the fulfillment element (if any).
- **Suspend:** Stop Oracle Provisioning from sending any orders to be provisioned and have all service orders placed in a queue until the fulfillment element becomes available.

### Adapter Scheduling

You schedule the use of adapters in one of three ways:

- **Immediate**: The adapter fulfills the request immediately, on the current date.
- **Future:** The adapter fulfills the request on a future date.
- **Periodic:** The adapter fulfills the request at periodic time intervals based on the supplied period.

### The Connection Manager

The Connection Manager provides the means to view, quickly modify, and manage the adapters connected to each fulfillment element. Adapters maintain connectivity to each fulfillment element during service provisioning.You can manage each connection for purposes such as load balancing, trouble shooting and maintenance.

You use the Connection Manager to accomplish the following tasks:

- View all the available fulfillment elements that have been configured in Oracle Provisioning
- View the total number of service order requests awaiting activation in the queue for a selected fulfillment element
- View the average time a service order request has been waiting in the queue
- View the number of adapters currently active for this fulfillment element
- View the date and time when the order entered the queue
- View the elapsed time a service order request has been in the queue
- View network problems and order flow to perform trouble shooting
- Monitor the number of adapters currently available
- Perform administrative activities on adapters
- Schedule activities for adapters

## Fulfillment Element Types in Oracle Provisioning

**Note:** Fulfillment element types are defined by system experts only.

Fulfillment elements in Oracle Provisioning are grouped into logical categories. A fulfillment element type can be any of the following:

- An operating system
- An equipment platform
- A Network Element Mediation (NEM) layer
- A piece of telecommunication-specific hardware

The table following lists examples of different fulfillment element types.

| Item | Fulfillment Element Type |
|---|---|
| Home Location Registration (HLR) unit | Network Element |
| Cisco server | Server |
| AT&T 5ESS switch | Telephony Switch |
| Nortel DMS200 switch | Telephony Switch |

### Parts to a Fulfillment Element Type

Any service order request provisioned through Oracle Provisioning requires that the type for that specific fulfillment element be defined in the system **before** the fulfillment element itself is defined.

To configure a fulfillment element type, you must define the following items:

- The fulfillment element type name and details
- The software generic properties for the fulfillment element type
- The attributes of the fulfillment element type

### Type Attributes

An attribute defines a property of a fulfillment element type. This can include all of the attributes with a connect / disconnect procedure. Attributes are also used to uniquely identify a fulfillment element.

Each fulfillment element inherits its attributes from its fulfillment element type. The value of an attribute is the default value defined at the level of the fulfillment element.

The attributes of an fulfillment element type defines its properties and behavior. These include the following:

- IP address
- Username

The following are additional attributes for a fulfillment element type:

- Passwords used in connection / disconnection procedures
- Software generics associated with the procedures

> **Note:** It is possible to configure multiple software versions (generics) for each fulfillment element type, each with a specific effective date.

All attributes are associated with a fulfillment element type or a fulfillment element.

- Each fulfillment element instance can have a different value.
- Each fulfillment element inherits attributes from its fulfillment element type.
- Adding attributes is performed at the fulfillment element type level.

> **WARNING:** Setting a value for an attribute overwrites the default value of that attribute specified at the fulfillment element type level.

### Fulfillment Element Type Attribute Default Values

Oracle Provisioning sets default values for the following attributes:

#### NE_CMD_TIMEOUT

When executing commands within a fulfillment element, the default set time-out is 120 seconds.

### NE_CMD_RETRY

The adapter retries the command numerous times before it times- out. A failure message is returned and the default value is set to zero retry.

### NE_CMD_WAIT

Waiting period (in seconds) before retrying commands within a fulfillment element. The default value is set to zero seconds.

### NE_NO_ACTIVITY_TIMEOUT

Some fulfillment elements may terminate the session, if there is no processing activity. The Service Delivery Platform allows you to set processing time to execute a command. This allows the connection between an adapter and the fulfillment element to remain active. The default processing time is set to 120 seconds.

### NE_DUMMY_CMD

The Service Delivery Platform allows you to send commands to a fulfillment element to maintain the active sessions with the adapter. The adapter sends a command five seconds before processing time-out (NE_NO_ACTIVITY_ TIMEOUT).

---

**Note:** If you want an carriage return command to be sent, use " " as the attribute value. The default command is " ".

---

### NE_CONNECT_RETRY_COUNT

Number of times the interactive adapter will try to reconnect automatically if the connection to the fulfillment element is dropped. The default value is zero.

### NE_CONNECT_WAIT

Elapsed time (in seconds) between attempts to reconnect. The Default value is zero.

### Fulfillment Elements in Oracle Provisioning

In the Oracle Service Delivery Platform, a fulfillment element is a unique physical entity. It has the following characteristics:

- It has a unique name on a carrier's network.
- It is a member of a unique Fulfillment Element type.

You must give each fulfillment element a unique name. This name is registered in the system during the configuration of the fulfillment element.

For example, if a telecommunications service provider has a router of a certain fulfillment element type (such as Cisco), then a logical name (Cisco_LA, or Cisco_NY, etc.) must be created for it during configuration. This helps to identify which particular Cisco router needs to be activated during provisioning.

### Definition Considerations

When configuring a fulfillment element, consider the following:

- A fulfillment element is **always** configured as an instance of a fulfillment element type.
- Attributes assigned to a fulfillment element type are inherited by the newly configured fulfillment element.
- Attribute values can be overwritten at the fulfillment element type level.

### Services

In Oracle Provisioning, a service is a telecommunication-related product, offered to the customer as an individual item or in bundles.

Examples of services include:

- Telephone services: Voice, data, videoconferencing, FAX, etc.
- Internet Protocol (IP) services: Web hosting, ecommerce, email, etc.

### Service Versions

It is possible to define more than one version of a service in Oracle Provisioning. You set the begin dates and end dates for each service/product version individually. However, only one version of the service is available to a given customer at one time.

## Work Items in Oracle Provisioning

Services provided by the carrier are fulfilled by a set of work items performed in a specific sequence within Oracle Provisioning. A work item, then, is a unit of work which is necessary to fulfil a service action.

- You assign to a work item the appropriate parameters, fulfillment actions and fulfillment procedures necessary for the provisioning process.
- Fulfillment actions are responsible for the actual provisioning of services at the fulfillment element level.
- As an alternative to work items, users can execute the fulfillment actions conditionally in a user-defined workflow. The workflow is defined according to pre-set conditions in a service order request.

### Work item Types

The available work item types are:

- Static: The list of fulfillment actions and their sequence are defined prior to runtime and do not change.
- **Dynamic:** The list of fulfillment actions and their sequence are determined at runtime by executing a shared procedure.
- **User-defined workflow:** The list of fulfillment actions and their sequence are determined at runtime by a predefined workflow that executes multiple procedures based on some pre-defined conditions.
- **User workflow procedure:** Sets the user-defined workflow that is to execute at runtime.

### Work Item Rules

Oracle Provisioning comes preseeded with a number of rules for adding, deleting, and modifying work items. They are used to determine dependencies between work items and to analyze the impact if a work item is modified or deleted.

### Packages

A service package contains at least two products / services which are sold together as a unit.

- A package may consist of either related or non-related services.
- All attributes and rules that apply to a service also apply to a service package.
- A service package must contain at least two mandatory products before it can be made available to customers. It can contain as many optional services as desired.

The dates of availability for the service package and the services / products within the package must be consistent. For example, you can not set the package available start date earlier than the earliest start date of all products in the package.

It is possible to add products / services to the package version. However, mutually exclusive products cannot be added to a package. In addition, products with dependencies (co-requisites) can not be added to a package unless the entire set is added.

During package creation, you must specify the minimum and maximum number of products the customer can order from a package that has optional products. For example, you may require that a customer pick at least three, but no more than five of eight possible services / products in the package.

**Note:** Once a product package has been defined and in the available state, it is not possible to modify or change the package. You must create a new version of the package to modify that package in any way.

### Notifications

The Notifications module is a utility which provides a graphical interface for monitoring service request orders that fail during provisioning.

If a service order request fails, then the provisioning workflow sends notifications about these failed service order requests to the Notifications utility, reporting provisioning problems at various phases of the provisioning process.

Users with sufficient privilege can review and correct the failed orders through manual intervention. The privileged user can, for example, perform the following tasks from within the Notifications utility:

- View order flowthrough.
- View workflow notifications for failed service order requests.
- Respond to a notification by assigning an error handling action to the service order request.
- Forward a notification to appropriate individuals or groups for resolution.
- Modify the service order request.
- View the workflow diagrams of order activities, using the Workflow Monitor.

If more details are necessary to asses the problem and resolve a failure, the privileged user can step through each of the failed notifications and view the actual provisioning commands sent by a particular service order request to a fulfillment element and the responses to those commands.

Based on the information contained in the notification messages received in the Notifications utility, the privileged user can follow-up with service order requests or workflow that failed during the provisioning process.

Follow-up actions include the following:

- Sending the order / workflow back to the process after modifying the order
- Sending orders to other personnel for escalation or resolution
- Stopping the process altogether

### The Notifications Utility

You view and access workflow notification messages through the Notifications utility.

You can view:

- All messages (both open and closed)
- Only the open notification messages.

You can query messages based on the following criteria:

- Status
- Receive date
- Order number
- Notification Type

**Note:** You can only search for orders which have their workflow started.

## Messaging in Oracle Provisioning

Oracle Provisioning obtains messages which are initiated from workflows to communicate with external systems. A message can be used to initiate a Start Billing event with a billing system. Workflow can be configured to wait for inbound messages or can branch out to another workflow based on the information passed.

### Message functions

The following are messages functions:

- Trigger a process
- Activate processes within an application
- Send application messages asynchronously using the Event Manager
- Receive application messages asynchronously using the Event Manager
- Handle incoming and outgoing messages and events using multiple queues and the Event Manager

### Message Definition

The following are ways to define a message:

- Message type: The message type defines a message either as a timer, an event, or a message.
  - Messages are used for communication between application systems.
  - Events are used to broadcast multi-cast state changes in business objects. Events are published to both external and internal application systems.
  - Timers are messages that have a time delay and a duration interval associated with them.
- Message for internal applications: Internal applications can register a PL/SQL callback procedure via the Event Publisher or through an API.
- Messages for external applications: External applications do not register callback procedures, but have adapters that relay the published event to the remote system. External applications can register for an event using the default subscribers screen.

### Message Queues

Messages can go to multiple queues. The following are some of the queues where messages can reside:

- **Inbound Message Queue**: Messages are sent to this queue for internal processing.
- **Outbound Message Queue**: Messages are sent to this queue that are enroute to peer systems.
- **Internal Event Queue**: Messages defined as Events ignore the queue name and the event is published to both the Outbound Message Queue and the Internal Event Queue.

The table following lists the messaging queues and their attributes.

| Queue Name | Service Name | Description |
|---|---|---|
| Inbound Message Queue | Message Server | Processes all incoming messages. |
| Internal Events Queue | Event Server | Events generated for internal consumption are enqueued on the internal events queue for speedy processing. |
| Outbound Message Queue | Communication Adapters | Dequeues messages from the Outbound Message queue and passes it to the peer system. |

### Outbound Message Structure

Outbound messages must have the following defined:

- Subscriber (system)
- Gateway name
- Adapter type

### Message Elements

Message elements in the message Studio can be marked either as mandatory or optional.

During message creation, the iMessage Studio automatically inserts the Message Code and MESSAGE as mandatory message elements in the Elements screen. In addition, the message code is defined as the root element. However, it is possible to define an element more than once in the structure.

Follow these guidelines during message creation:

- Message element names must follow PL/SQL naming conventions.
- The internal name for all message elements must be defined in upper case, replacing any spaces with an underscore.

**Warning: Never delete the root element of a message.**

- Message elements marked as parameters automatically use the default value specified.
- Message elements marked as parameters are used to derive values of remaining message elements.
- Message elements marked as parameters are not passed in the content of the message body.
- Message elements marked as parameters generate CREATE_MSG(), SEND() and PUBLISH() procedures with the specified message element as a parameter defaulted to the message specified.

### Mandatory Elements

- Message elements can be marked as mandatory or optional.
- Upon receiving an incoming message, the validation logic checks all mandatory message elements to ensure that values are in the correct data format.

### Message Processing Logic

The Oracle Provisioning Event Manager handles all incoming messages and events. There are four possible ways to process a message. The table following lists them.

***Message Processing Logic Types***

| Type | Process | Description |
|---|---|---|
| Default Process Logic | DEFAULT_PROCESS() | Automatically executed by the Event Manager if no application has registered for the message. |
| Validate Logic | VALIDATE() | Automatically executed by the Event Manager on the newly arrived messages.<br>The VALIDATE() procedure provides a hook to include business specific validation. |
| Incoming Message Process Logic | PROCESS() | Executed by the Event manager before delivering the message to the callback procedure of the registered application.<br>The PROCESS() procedure also provides a hook to include any application logic. |
| Outgoing Message Process Logic | SEND() | Executed by the Event Manager before publishing a message to the Outgoing Queue for delivery.<br>These logic procedures are not generated automatically, they are user-defined and the code is executed as part of the SEND() procedure. |

### Processing Failure

The Event Manager does not deliver a message to any registered applications if an error is detected during the execution of the processing logic. Nor does the Event Manager process and deliver a message if the message validation process failed. If an error occurs, each resulting error code and error message are logged into the system log.

### Message Processing Logic - Guidelines

Use the following guidelines during creation of your message processing logic procedures.

1. If no application logic is specified, the procedure is created with a single **NULL;** statement
2. Any user defined data which is stored as part of the registration process is passed to the procedure in the parameter **p_process_reference**.
3. The parameter **p_msg_text** is the XML message
4. Element values can be obtained using the following procedure:

   xnp_xml_utils.decode(). xnp_xml_utils.decode

   This procedure works only if a message has no repeating elements.
5. User defined logic is required to return any error code and error message. This information must be passed using the parameters:
   - x_error_code
   - x_error_message
6. Any non zero value in x_error_code is considered to be an error.

## Timers in Oracle Provisioning

In any reasonably complex business process, system and user generated business events are continually occurring. These business events are not isolated, but are often required as triggers to further business processes. Such business processes may have time constraints.

Timers in Oracle Provisioning are used to handle the events or processes that must occur at specified time intervals during the course of implementing complex business processes.

Timers in Oracle Provisioning are implemented using the Oracle Advanced Queuing mechanism, and the message builder features of the Oracle Service Delivery Platform.

### Timer Definition

A timer is a message that consists of the following two elements:

- **Delay:** Represents the amount of time to wait before starting a timer.
- **Interval:** Represents the wait time for a timer.

### Timer Categorization

Timers are divided into three categories in Oracle Provisioning:

- Activity Timer: The timer is associated with a particular activity within a workflow.
- **Process Timer:** The timer is associated with an entire workflow process.
- **Message Timer:** The timer is associated with a message.

### Timer Usage in Oracle Provisioning

You use the iMessage Studio Message Framework to create the different types of timers that you use in Oracle Provisioning.

### Timer Types

Oracle Provisioning comes with three different timer types. The table following describes each type.

| Timer Type | Description |
|---|---|
| Activity Timer | The timer is associated with a particular activity within a workflow. |
| Process Timer | The timer is associated with an entire workflow process. |
| Message Timer | The timer is associated with a message. |

### Timer Elements

Timer messages requires two mandatory elements:

- Delay
- Interval

Additional optional elements can also be related to a product type, customer category, or the service level agreement based on the business requirements of the user.

### Timer Data

It is possible to set the delay and interval timer elements in several different ways:

- The user provides the data source for the timer elements. This feature is useful in retrieving the timer information from other timer configuration tables.
- A stored procedure retrieves the timer delay and interval information.

If desired, you can also set the processing logic that occurs when the timer expires. The processing logic may include jeopardy management or error processing features when the timer expires.

### Event for Timer or Message Acknowledgment

The Waiting For Acknowledgments activity subscribes to events. An event may be configured to include and Acknowledgment for the Message sent out and a Timer. Depending on the message received, the workflow progresses.

For example.,

We may have an event group call MSG_ACK_TIMER_1 that consists of Timer 1 and Ack. Depending on Timer 1 or Ack being received, the workflow progresses.

### Timer Testing

- You may also specify that a test message be sent to test that the Timer message has been correctly configured.Define a connection management schedule as needed

# User Roles

## Using the Flowthrough Manager

It is possible to view status details and other details of a service order while it is in one of the following phases:

- Inquiry (If a porting record was created during the inquiry.)
- Ordering
- Active
- Old

To view the porting summary, perform the following steps.

### Prerequisites

None

### Steps

1. In the Navigator, choose **Operations > Flowthrough Manager**.

   If an order has not yet been referenced, the Find Orders dialog box displays initially.

2. Enter a value to use as the search criteria in the Find Orders dialog box.

   You can search for an order on several criteria. If you know the order ID or order number, then simply enter this information and click **Show Order**. Alternatively, you can enter one or more of the other criteria to search for the order.

   The Flowthrough Manager displays the found order (with the order number in the left-hand list box). All of the work items associated with this order are displayed in the form of leaf nodes. If necessary, expand the list of work items by clicking the plus sign next to the order number.

3. Select the Order Header tab to view the basic information associated with this order.

   The current provisioning status of the order displays in the Order Status field. Each provisioning order is in one of a number of states and within a state it may be in one of a number of statuses. Provisioning states and statuses are pre-seeded by the application.

   To view information about the Notifications, Event Log, Workflow and Parameters associated with this order, click the appropriate button at the bottom of the screen.

4. Select the Order Line tab to view information relating to the current provisioning status and associated parameters for this order.

   - To view information about the workflow associated with this order, click **Workflow**.
   - To view information about the parameters associated with the order line, click **Parameters**.

5. Select the **Work Items** tab to view status and date information relating to work items associated with this order (if any).

   - To view status and state information, select the Status sub-tab.
   - To view date-related information, select the Date sub-tab.
   - To view information about the parameters associated with a work item, select it and click **Parameters**.
   - To view information about the workflow associated with this work item, click **Workflow**.

6. Select the Fulfillment Actions tab to view information relating to fulfillment actions associated with this work item and the fulfillment element to which each action is being applied.

   - To view status-related information, select the Status sub-tab.
   - To view date-related information, select the Date sub-tab.
   - To view Parameter, Audit Trail, and Workflow information for a fulfillment action, select a fulfillment action (if any exist for this order) and click the appropriate button at the bottom of the screen.

7. Click **Cancel** to exit the Order Flowthrough screen.

## Managing Notifications

From the Notification Inbox, you can take various actions to deal with workflow notification. For example, you can perform the following tasks:

- Viewing notifications
- Adding comments to a notification message
- Responding to a message notification
- Forwarding a message notification

### Viewing Notifications

To view the existing notification messages in the system, perform the following steps.

#### Prerequisites

None

#### Steps

1. In the Navigator, choose **Operations > Notifications**.
2. Click **View Open** to view only those notification messages that are currently open.
3. Click **View All** to view notification messages, regardless of status.
4. Select a notification from the list at the left.

   The Notification Message Details window opens.
5. Close the window after viewing the information.

### Adding Comments to a Notification Message

To add a comment to a notification message, perform the following steps.

#### Prerequisites

The message must exist for this option to be available.

### Steps

1. In the Navigator, choose **Operations > Notifications**.
2. Select a notification from the list at the left.
3. Click **View Details**.

   The Notification Message Details window opens.
4. Click **Add Comment** to launch the Notification Comment window.
5. Enter your comments about the notification in the space provided.
6. Close the window.
7. Save your changes.

### Responding to a Notification Message

To respond to a notification message, perform the following steps. Responding to a notification changes its status from Open to Closed.

### Prerequisites

The status of the message must be **Open** for this option to be available.

### Steps

1. In the Navigator, choose **Operations > Notifications**.
2. Select a notification from the list at the left.
3. Click **View Details**.

   The Notification Message Details window opens.
4. Click **Respond** to launch the Notification Response window.
5. Select one of the following responses from the drop down menu.
   - **Abort**: Terminate the process and close the order.
   - **Resolved**: Ignore the reported causes of failure and continue processing the order.
   - **Retry**: Retry the execution of the failed order.
6. Click **OK** to close the window and initiate the action.

**Note:** The **Close Notification** response is available for all of the information types. The status of a notification stays **Open** until you respond to it.

### Forwarding a Message Notification

To forward the notification to a third party, perform the following steps.

### Prerequisites

The status of the message must be **Open** for this option to be available.

### Steps

1. In the Navigator, choose **Operations > Notifications**.
2. Select a notification from the list at the left.
3. Click **View Details**.

   The Notification Message Details window opens.
4. Click **Forward** to launch the Notification Response window.
5. Select another user or group from the drop-down menu.
6. Click **OK** to close the window and reassign the message.

**Note:** When a notification is reassigned, it is marked as **Closed**.

## Resubmitting an Order

You use the Resubmission Utility to re-run orders that have been previously submitted to the Service Delivery Platform provisioning engine. This is useful, for example, if a network switch fails, or experiences service disruption that causes a provisioning order to fail. The order, however, has not been lost by the system, and can be resubmitted when the unit becomes available.

For example, a particular fulfillment element fails at 3:02 a.m. on a particular date. It remains unavailable for service until 5:20 a.m. the same day. This fulfillment element was to be used to provision a number of orders during that span of time, but as it was unavailable, the orders failed to complete. After the fulfillment element (or its replacement) becomes available, it is necessary to resubmit all orders that were to be provisioned by that particular fulfillment element during the window of time from 3:02 to 5:20 a.m. The order information is not lost, and can be re-used to resubmit the original provisioning orders.

You use the Resubmission Utility to perform the following tasks.

- Creating a new resubmission job
- Verifying the status of a resubmission job

### Creating a New Resubmission Job

To resubmit all orders that were to be provisioned by a specific fulfillment element during a span of time (but were not provisioned), perform the following steps.

#### Prerequisites

None

#### Steps

1. In the Navigator, choose **Operations > Order Resubmission**.

   The Order Resubmission Jobs form window opens. Previously defined resubmission jobs display under Job ID (Fulfillment Element) at the left-hand side of the screen.

2. Click New.

   The New Job Window opens. You use this window to create a new resubmission job that targets a specific fulfillment element during a specified time window.

3. Enter the name of the fulfillment element for which you wish to resubmit orders.

   Chose a fulfillment element from the List of Values (LOVs).

4. Enter the exact date and time at which the fulfillment element became unavailable in the Outage Start Date field. And, if desired, enter the exact date and time that the fulfillment element became available again.

5. Enter the number of adapters that this affects in the No. of Adapter field.

6. Click **Submit**.

   The job displays in the Job ID (Fulfillment Element) window. You can track the progress of order resubmission process through the Job ID.

7. Click **Cancel** to close this window.

### Verifying the Status of a Resubmission Job

To view the status or details of a resubmission job, perform the following steps.

### Prerequisites

None

### Steps

1. In the Navigator, choose **Operations > Order Resubmission**.

   The Order Resubmission Jobs form window opens. Previously defined resubmission jobs display under Job ID (Fulfillment Element) at the left-hand side of the screen.

2. Select the Job ID from the list at the left.

   The pertinent information for the resubmission job displays.

3. Click **Order List**.

   The Resubmitted Order Information window opens. This window displays information about each individual order.

4. Click **Cancel** to close the Resubmitted Order Information window.

5. Click **Cancel** to close the Order Resubmission Jobs window.

## Managing the System Queues

You perform a number of tasks with system queues. These include:

- Viewing summary information about queues
- Finding an order in the queue
- Viewing errors in a queue
- Starting up a queue
- Shutting down a queue
- Suspending a queue
- Resuming queue operation
- Viewing XML messages

### Viewing Summary Information about Queues

The Queue Console displays the following summary information about all of the system queues.

- State (Enabled, Shutdown, Suspended)
- Number of entries in the queue
- Date that the last entry in the queue occurred
- Number of dequeue processors that are registered
- Number of processors that are currently running for the queue

To display this information, perform the following steps.

#### Prerequisites

None

#### Steps

1. In the Navigator, choose **Administration > Queue Console**.

   The Queue Console opens.

2. Select a queue.

   The View Details button is enabled if there are entries in the queue, otherwise it is grayed out.

3. Click **View Details** to view additional information about the entries in a queue.

   A window opens that displays the entries for the queue.

4. Close this window and return to the Queue Console by clicking the **X** in the upper right-hand corner of the window.

### Finding an Order in the Queue

To search for an order that is currently in the queue, perform the following steps.

### Prerequisites

None

### Steps

1. In the Navigator, choose **Administration > Queue Console**.

   The Queue Console opens.

2. Select a queue.

   The View Details button is enabled if there are entries in the queue, otherwise it is grayed out.

3. Click **View Details**.

   A window opens that displays the entries for the queue.

4. (Optional) Select **View > Query By Example > Enter** from the View menu on the toolbar to find a specific order in the queue, or to find entries that match a search criteria.

5. (Optional) Enter the search criteria in the provided fields.

6. Select **View > Query By Example > Run** from the View menu on the toolbar.

   If one, or more records are found that match the search criteria, then the information displays.

7. Select **View > Query By Example > Cancel** from the View menu on the toolbar to close the window and return to the Queue Console.

---

**Note:** If there are a large number of orders in the queue, this action may take several seconds to complete.

---

### Viewing Errors in a Queue

A queue becomes **disabled** if an error is detected while processing orders. You mange this problem by performing the following actions:

- Viewing the cause of the error that caused this state to occur
- Correcting the problem the caused the error to occur
- Re-starting the queue

To view errors within a queue, perform the following steps.

#### Prerequisites

The queue must be in Disabled mode for you to perform this operation.

#### Steps

1. In the Navigator, choose **Administration > Queue Console**.

   The Queue Console opens.

2. Select a queue that is listed as Disabled.

   The View Errors button is enabled only if the queue is disabled.

3. Click **View Errors**.

   The Errors window opens and displays all the errors associated with the queue.

4. Click **OK** to close the window and exit.

> **Note:** Orders remain in the queue until the queue is shutdown and started again.

### Starting Up a Queue

To start all processors of the queue, perform the following steps.

#### Prerequisites

The queue must be in Shutdown mode for this option to be available.

### Steps

1. In the Navigator, choose **Administration > Queue Console**.

   The Queue Console opens.

2. Select a queue.

   If the queue is in Shutdown mode, the Startup button is enabled.

3. Click **Startup**.

   The Queue Status changes to Enabled.

## Shutting Down a Queue

To shut down all processors of the queue, perform the following steps.

### Prerequisites

The queue must be in either the Enabled or Disabled mode for this option to be available.

### Steps

1. In the Navigator, choose **Administration > Queue Console**.

   The Queue Console opens.

2. Select a queue that is either in the Enabled or the Disabled mode.

3. Click **Shutdown**.

   The Queue Status changes to Shutdown.

## Suspending a Queue

To suspend processing of orders in the queue, perform the following steps.

### Prerequisites

The queue must be in either the Enabled or Disabled mode for this option to be available.

### Steps

1. In the Navigator, choose **Administration > Queue Console**.

   The Queue Console opens.

2. Select a queue that is either in the Enabled or the Shutdown mode.

   If the queue is in Enabled or Shutdown mode, the Suspend button is enabled.

3. Click **Suspend**.

   The Queue Status changes to Disabled.

### Resuming Queue Operation

To resume processing orders in the queue, perform the following steps.

### Prerequisites

The queue must be in the Disabled mode for this option to be available.

### Steps

1. In the Navigator, choose **Administration > Queue Console**.

   The Queue Console opens.

2. Select a queue that is in the Disabled mode.

   If the queue is in Disabled mode, the Resume button is enabled.

3. Click **Resume**.

   The Queue Status changes to Enabled.

### Viewing XML Messages

To view the XML content for a message that is currently in the queue, perform the following steps.

### Prerequisites

To view XML message content, you must select either the Outbound Messages queue, the Inbound Messages queue, or the Event queue, **and** you must have an XML-enabled browser available to display XML messages.

### Steps

1. In the Navigator, choose **Administration > Queue Console**.

   The Queue Console opens.

2. Select a queue.

   The View Details button is enabled if there are entries in the queue, otherwise it is grayed out.

3. Click **View Details**.

   A window opens that displays the entries for the queue.

4. Select a message from the list at the left.

5. Click **View XML**.

   If available, an XML-enabled browser window opens and displays the XML message.

## Other Considerations

In order to efficiently complete an implementation, you should do the following:

- Provide the telecommunications service provider with a report highlighting your findings, conclusions and recommendations on implementations time frame.
- If any processes are not defined, escalate this as an issue for not being able to define properly the business process Workflow in Oracle Provisioning.
- Present the report to the appropriate management.
- Until you have gathered all the information and the right resources for the project DO NOT PROMISE A DELIVERY OF PRODUCTION READY SOFTWARE TO THE CUSTOMER.
- Bring recommendations to the Customer, action items to resolve issues, concerns, and directions and associate the next steps.

# Typical Release Dependencies

Oracle Provisioning 11i is based on the foundation of the Oracle installed base that includes:

- Common Data Model
- Common Schema
- Technical StackOracle Product Name

# Setting Up Oracle Provisioning

## Information Gathering

### Required Information

Implementations for telecommunications service providers cam be complex or simple. In many cases it will take simple identification of the implementation's scope.

The following needs to be identified when planning an implementation for Oracle Provisioning:

- Number of services
- Actions required for each service
- Network elements needed to be provisioned for each service
- Number of processes (to include exception handling)
- Anticipated future items to be implemented
- Time and resources needed to complete the project (done by Oracle Consulting)

### Gathering Information

The following tasks are involved in planning an implementation for Oracle Provisioning:

### Planning Information Gathering

While the direct approach (interviews with the telecommunications service provider) my be the primary way to gather information, do not overlook other methods. Some of these may be to gather information by:

- Observing telecommunication service provider operations
- Talking to telecommunication service provider customers
- Interviewing the telecommunication service provider who perform the processes

To help you gather information, do the following:

- List information needed
- Identify sources for collecting information (keeping in mind that there may be more than one source)
- Correlating gathered information with appropriate processes and tasks for implementation and process building

### Asking questions

The following are examples of questions you can use when gathering information:

- How is a Service Order Request captured initially?
- What is the data captured and where is it coming from?
- Which schema will the data reside - in the Order Entry, in Oracle Provisioning or both? Or will the data and schema depend on certain parameters only?
- Must the data be derived from external sources (e.g., outside Service Providers, external systems outside of SDP, etc.)?
- What is the syntax for each service to be activated within a Network Element?
- How will the customer handle exceptions, errors, failures?
- Is there a second path for activation if the first path fails, who needs to be updated?

### Implementation Tasks

- Form an implementation team
- Bring resources from Oracle Consulting to help implement the project
- Identify the Customer's needs and submit a plan with guidelines and time frames for project completion
- Submit a plan for identifying and for resolving project roadblocks
- Identify overall existing Customer's business processes and the new business processes which will be implemented by Oracle Consulting
- Identify the list of telecommunication service provider's products and services
- List telecommunication service provider's priorities for implementation
- Identify the overall network infrastructure for telecommunication service provider's products and services.
- Identify the number of subscribers
- Correlate telecommunications service provider's products and services with their subscribers
- List the issues the telecommunications service provider needs to address
- Identify interface requirements needed for compatibility with other systems
- Identify the incompatible systems

### Sources of Information

For a successful implementation there will be numerous information items you will need to gather. Some of these include:

- Collect obvious information
- Uncover hidden information
- Identify issues
- List concerns
- Address needs
- Determine direction (telecommunications service provider's)

While there are many sources for information, the following telecommunications service provider's department are a good start to find information items:

- Service Provisioning
- Order Processing
- Network Management
- Engineering
- Information Technology
- Customer Service

### Information Gathering Considerations

The following are items to consider when gathering information:

- Use the "right person" to obtain information from at the telecommunications service provider's site. Do not select interviews by title. Try to find and interview the person closest to the operation. This may include:
  - Network / Operations Engineers
  - Process Engineers
  - Marketing Associates
  - Order Entry Analysts.
- Interview and debrief the telecommunications service provider's upper management. This will help you determine some general corporate issues and keep them informed about the projects's progress.

- Identify the current number of products and services to be deployed by the telecommunications service providers.
- List number of network elements which need to be provisioned in order to activated the products of services.
- Identify future plans (growth) for products, services, facilities, and network elements.
- Verify syntax /commands that are used to provision each service within a network element.
- List the actions needed for each product, services, and network elements.
- Identify the number of activation procedures to be written for new product and service provisioning.
- Identify the number of Workflow (business processes and exception handlings) to be written.

## Provisioning Activities Planning

The user should be able to define business rules in the following PL/SQL procedures for the selection of work item to execute for a given service, the selection of fulfillment actions to execute for a given work item, and the selection of Oracle Workflow process to execute for a given work item.

### Service to Work Item Dynamic Mapping

During order capturing time, SDP will need to determine which work items to execute for a given service. Sometimes the list of work items to execute for a service is fixed and can be defined by an SDP user during configuration time through the SDP Configurator. Sometimes the list of work items to run can vary depends on certain parameter values in the order during runtime. For example, a company can have a Nortel network and a Lucent network on different geographical areas and the work items for the same service can vary depends on which network it will be provisioned on. SDP provides a solution to such problem by allowing the user to define business logic in the service to work item mapping procedure, where the user can instruct SDP to execute different set of work items base on the actual value of a service parameter, the area code for example. An example will be given later in this chapter.

### Procedure Specification

When defining the service to work item mapping procedure through the Configurator, SDP will automatically generate the following procedure specification:

```
procedure <name of the procedure>(
         p_order_id IN NUMBER,
        p_line_item_id IN NUMBER,
p_return_code OUT NUMBER,
       p_error_description OUT VARCHAR2)
```

The user can not change the procedure specification. However, the user can refer to those procedure parameters as desire anywhere in the procedure body.

## Work Item to Fulfillment Action Dynamic Mapping

During work item execution, SDP will need to determine which provisioning activities, namely fulfillment actions, need to be executed for a given provisioning job, namely work item. Sometimes the list of fulfillment actions (FAs) to execute for a work item is fixed and can be defined by an SDP user during configuration time through the SDP Configurator. Sometimes the list of FAs to run can vary depends on certain parameter value in the order during runtime. For example, a work item PROVISION_CARRIER_SWITCH may invoke different FAs depends on the long distance carrier the customer chooses. SDP provides a solution to such problem by allowing the user to define business logic in the work item to FA mapping procedure, where the user can instruct SDP to execute different set of FAs base on the actual value of a service parameter, the carrier code for example. An example will be given later in this chapter.

### Procedure Specification

When defining the work item to FA mapping procedure through the Configurator, SDP will automatically generate the following procedure specification:

```
procedure <name of the procedure>(
         p_order_id IN NUMBER,
        p_line_item_id  IN NUMBER,
        p_wi_instance_id IN NUMBER,
p_return_code OUT NUMBER,
       p_error_description OUT VARCHAR2)
```

The user can not change the procedure specification. However, the user can refer to those procedure parameters as desire anywhere in the procedure body.

## Work Item Workflow Execution Procedure

Sometimes the work item execution can be a really complex process. For example, when a customer orders a T1 circuit, the carrier may need to get a circuit design first, and then provision the appropriate fulfillment elements accordingly. In such case SDP will not be able to determine which FA to execute for this particular work item at the beginning of the provisioning process due to the fact that, the next FA execution is solely base on the execution result of the previous FA. SDP addresses this problem by allowing the user to associate an Oracle Workflow with the work item, where the users can define their own business process flow for the work item execution. In some practice the user may want to define multiple workflow processes for the same work item base on their network implementation. For example, they may have different network configuration on different geographical areas which in turn may require different business process to execute for the same work item. The user can then implement their workflow selection rules in the work item workflow execution procedure, which SDP will use to select the appropriate workflow process base on the order information at runtime.

### Procedure Specification

When defining the work item workflow execution procedure through the Configurator, SDP will automatically generate the following procedure specification:

```
procedure <name of the procedure>(
      p_order_id IN NUMBER,
      p_line_item_id  IN NUMBER,
      p_wi_instance_id IN NUMBER,
      p_wf_item_type OUT varchar2,
      p_wf_item_key  OUT varchar2,
      p_wf_process_name  OUT varchar2,
      p_reurn_code OUT NUMBER,
      p_error_description OUT VARCHAR2)
```

The user can not change the procedure specification. However, the user can refer to those procedure parameters as desire anywhere in the procedure body.

## Provisioning Activities Execution

The user should be able to define business rules in the following PL/SQL procedures for evaluating the parameter value at runtime, identifying which fulfillment element to run against for an FA, and provisioning a fulfillment element for an FA.

### Work Item Parameter Evaluation Procedure

This procedure allows the user to evaluate a work item parameter value at runtime. For example, the user can use this procedure to retrieve value for certain provisioning specific parameter from systems such as network inventory database, or change the format of the work item parameter value to best fit their business need.

### Procedure Specification

When defining work item parameter evaluation procedure through the Configurator, SDP will automatically generate the following procedure specification:

```
procedure <name of the procedure>(
     p_order_id IN NUMBER,
     p_line_item_id    IN NUMBER,
      p_wi_instance_id IN NUMBER,
      p_param_valIN Varchar2,
      p_param_ref_valIN Varchar2,
      p_param_eval_val  OUT VARCHAR2,
     p_param_eval_ref_val  OUT Varchar2,
     p_return_codeOUT NUMBER,
     p_error_description  OUT VARCHAR2)
```

The user can not change the procedure specification. However, the user can refer to those procedure parameters as desire anywhere in the procedure body.

## Fulfillment Action Parameter Evaluation Procedure

This procedure allows the user to evaluate a fulfillment action parameter value at runtime. For example, the user can use this procedure to retrieve value for certain provisioning specific parameter from systems such as network inventory database, or change the format of the work item parameter value to best fit their business need.

### Procedure Specification

When defining work item parameter evaluation procedure through the Configurator, SDP will automatically generate the following procedure specification:

```
procedure <name of the procedure>(
      p_order_id IN NUMBER,
      p_line_item_id IN NUMBER,
      p_wi_instance_id IN NUMBER,
      p_fa_instance_id IN NUMBER,
      p_param_valIN Varchar2,
      p_param_ref_valIN Varchar2,
      p_param_eval_val  OUT VARCHAR2,
     p_param_eval_ref_val  OUT Varchar2,
     p_return_codeOUT NUMBER,
     p_error_description  OUT VARCHAR2)
```

The user can not change the procedure specification. However, the user can refer to those procedure parameters as desire anywhere in the procedure body.

## Fulfillment Element Routing Procedure

This procedure allows the user to identify which fulfillment element to provision for a fulfillment action base on the order information at runtime.

### Procedure Specification

When defining fulfillment element routing procedure through the Configurator, SDP will automatically generate the following procedure specification:

```
procedure <name of the procedure>(
      p_order_id IN NUMBER,
      p_line_item_id      IN NUMBER,
      p_wi_instance_id IN NUMBER,
      p_fa_instance_id IN NUMBER,
     p_fe_name  OUT Varchar2,
     p_return_codeOUT NUMBER,
     p_error_description  OUT VARCHAR2)
```

The user can not change the procedure specification. However, the user can refer to those procedure parameters as desire anywhere in the procedure body.

## Fulfillment Procedure

This procedure allows the user to define the provisioning logic on a particular fulfillment element type for a fulfillment action. The user will be able to use a set of macros provided by SDP to send the appropriate commands or message to the fulfillment element, examine the response from the fulfillment element and determine the next command or message accordingly.

### Procedure Specification

When defining work item parameter evaluation procedure through the Configurator, SDP will automatically generate the following procedure specification:

```
procedure <name of the procedure>(
      p_order_id IN NUMBER,
      p_line_item_id      IN NUMBER,
      p_wi_instance_id IN NUMBER,
      p_fa_instance_id IN NUMBER,
      p_channel_nameIN Varchar2,
      p_fe_nameIN VARCHAR2,
      p_fa_item_type   IN VARCHAR2,
      p_fa_item_key    IN VARCHAR2,
     sdp_internal_err_codeOUT NUMBER,
      sdp_internal_err_str OUT VARCHAR2)
```

The user can not change the procedure specification. However, the user can refer to those procedure parameters as desire anywhere in the procedure body.

### Network Communication Process

Sometimes the communication between SDP and the fulfillment element may require SDP to open a connection to the equipment before passing messages or commands to the FE. This chapter will describe how to implement the connection logic in the connect / disconnect procedure which will be executed by SDP when a communication channel is established between SDP and the fulfillment element.

### Connect Procedure

This procedure allows the user to instruct SDP how to establish a connection to a fulfillment element.

#### Procedure Specification

When defining connect procedure through the Configurator, SDP will automatically generate the following procedure specification:

```
procedure <name of the procedure>(
 p_fe_name IN Varchar2,
p_channel_nameIN Varchar2,
p_return_code IN OUT NUMBER,
     p_error_description IN OUT VARCHAR2
)
```

The user can not change the procedure specification. However, the user can refer to those procedure parameters as desire anywhere in the procedure body.

### Disconnect Procedure

This procedure allows the user to instruct SDP how to close a connection from a fulfillment element.

#### Procedure Specification

When defining connect procedure through the Configurator, SDP will automatically generate the following procedure specification:

```
procedure <name of the procedure>(
 p_fe_name IN Varchar2,
p_channel_nameIN Varchar2,
p_return_code IN OUT NUMBER,
     p_error_description IN OUT VARCHAR2)
```

The user can not change the procedure specification. However, the user can refer to those procedure parameters as desire anywhere in the procedure body.

## Managing Network Connections

You use the Connection Manager to manage all types of fulfillment elements and their network adapters. The Connection Manager is used to control network connections.

You perform the following tasks using the Connection Manager:

- Monitoring network connections
- Defining a new adapter
- Redefining an adapter
- Scheduling an adapter
- Deleting an adapter
- Viewing adapter request errors
- Issuing a disconnect request
- Issuing a connect request
- Issuing a suspend request
- Issuing a resume operation request

### Monitoring Network Connections

To view the network associated with a fulfillment element, perform the following steps.

### Prerequisites

None

### Steps

1. In the Navigator, choose **Administration > Connection Manager**.

   The Connection Manager opens.

2. Click Enter a filtering value if you wish to parse the list of fulfillment elements that display in the left-hand frame.

   Conversely, erase the filtered value entry to view all defined fulfillment elements.

3. Select a fulfillment element from the list.

4. Select the Adapters tab to view information about network adapters.

5. Select the adapter type to view from the Usage drop-down list. Your choices are:

   - All Types
   - Messaging
   - Order resubmission
   - Provisioning
   - Testing

6. Choose one of the following:

   - Chose the Adapters sub-tab to view information about the current status of an adapter.
   - Chose the Modes sub-tab to view information about Startup and Debug modes for an adapter.
   - Chose the Requests sub-tab to view to view information about how the adapter has been scheduled.

7. Click the "**x**" in the right-hand corner of the form window to close the form window.

### Defining a New Adapter

To define a new adapter in the system, perform the following steps.

### Prerequisites

None

### Steps

1. In the Navigator, choose **Administration > Connection Manager**.
2. Select the Adapters tab.
3. Click **New...** to open the Adapter Properties dialog box.
4. Enter a name for r the new adapter.
5. Select a value from the Usage drop-down list.
6. Select a value from the Startup Mode drop-down list.
7. Select a value from the Debug Mode drop-down list of values.
8. Click **OK** to define the new adapter and close the dialog box.

   The new adapter now displays in the list of adapters.

---

**WARNING: You can start a new adapter only if it does not exceed the maximum number of connections allowed (or configured) for that specific fulfillment element**.

---

### Redefining an Adapter

To edit an adapter definition, perform the following steps.

### Prerequisites

The Current Status field for the adapter must display Shutdown for this option to be available.

### Steps

1. In the Navigator, choose **Administration > Connection Manager**.
2. Select the adapter from the list at the left.
3. Select the Adapters tab.
4. Click **Edit**.
5. The Adapter Properties dialog box displays.
6. Make changes as necessary.
7. Click **OK** to redefine the adapter and close the dialog box.

### Scheduling an Adapter

To schedule a request for an adapter, perform the following steps.

### Prerequisites

You can not schedule a request prior to the current date and time.

### Steps

1. In the Navigator, choose **Administration > Connection Manager**.
2. Select the Adapters tab.
3. Click **Schedule...** to open the Adapter Request dialog box.
4. Chose a value from the Request Type drop-down list of values.
5. Chose a date and time for the request to execute the request. (If desired, open the Calendar function by clicking the [...] at the edge of the Date/Time field.)
   - To issue the request immediately, enter the current date and time.
   - To postpone the execution of the request, enter a future date.
6. Enter the period (in days) that the adapter request is to repeat by entering a value in the Repeat Every (days) field.
   - To schedule a request only once, leave this field blank.
7. Click **OK** to schedule the request and close the dialog box.

---

**Note:** To delete a scheduled request, select it and click **Delete**.

---

### Deleting an Adapter

To delete an adapter from the system, perform the following steps.

#### Prerequisites

The Current Status field for the adapter must display Shutdown for this option to be available.

#### Steps

1. In the Navigator, choose **Administration > Connection Manager**.
2. Select the adapter to delete from the list of adapters.
3. Click **Delete**.

   The adapter no longer displays in the list of adapters.

### Viewing Adapter Request Errors

If an adapter request causes an error condition, the Current Status field for that adapter displays Error.

To view details of the error condition, perform the following steps.

#### Prerequisites

The Current Status field for that adapter must display Error for this option to be available.

#### Steps

1. In the Navigator, choose **Administration > Connection Manager**.
2. Select the adapter from the list of adapters at the left.
3. Select the Adapters tab.
4. Click **Show Error**.

   The time the error message occurred and the text of the error message display.
5. Click **OK** to close the Error Message window.

### Issuing a Disconnect Request

To issue a disconnect request to an adapter, perform the following steps.

#### Prerequisites

The Current Status field for that adapter must display Idle for this option to be available.

#### Steps

1. In the Navigator, choose **Administration > Connection Manager**.
1. Select the adapter from the list of adapters at the left.
2. Select the Adapters tab.
3. Click **Disconnect**.

   The Current Status field for this adapter now displays Disconnected.

### Issuing a Connect Request

To issue a connect request to an adapter, perform the following steps.

#### Prerequisites

The Current Status field for the adapter must display Disconnected for this option to be available.

#### Steps

1. In the Navigator, choose **Administration > Connection Manager**.
2. Select the adapter from the list of adapters at the left.
3. Select the Adapters tab.
4. Click **Connect**.

   The Current Status field for this adapter now displays Idle.

### Issuing a Suspend Request

To issue a suspend request to an adapter, perform the following steps.

#### Prerequisites

The Current Status field must display Idle for this option to be available.

#### Steps

1. In the Navigator, choose **Administration > Connection Manager**.
2. Select the adapter from the list of adapters at the left.
3. Select the Adapters tab.
4. Click **Suspend**.

   The Current Status field for this adapter now displays Suspended.

### Issuing a Resume Operation Request

To issue a resume request to an adapter, perform the following steps.

#### Prerequisites

The Current Status field must display Suspended for this option to be available.

#### Steps

1. In the Navigator, choose **Administration > Connection Manager**.
2. Select the adapter from the list of adapters at the left.
3. Select the Adapters tab.
4. Click **Resume**.

   The Current Status field for this adapter now displays Idle.

## Managing Services

You perform a number of tasks with services. These include:

- Defining a new service
- Associating actions with services
- Associating work items with service actions
- Editing an existing action
- Deleting a service

For the steps and tasks associated with setting up a service, see Configuring Services for an overview of the processes involved.

### Configuring Services

The tasks necessary to configure a service are described in the following steps.

### Prerequisites

None

### Steps

In configuring services, you must define all of the following items.

| | Item | See |
|---|---|---|
| 1. | The details of the service (for example, version, description and activation dates) | Defining a New Service |
| 2. | The actions available for a specific service and version | Associating Actions with a Service |
| 3. | The work items used by the service actions | Associating Work Items with Service Actions |

### Defining a New Service

Services are also referred to as products or service offerings. In each case, however, they represent services deployed by a service provider who delivers, activates, maintains and bills a subscriber for it.

To define a new service, perform the following steps.

### Prerequisites

None

### Steps

1. In the Navigator, choose **Setup > Service Definitions > Services**.
2. Select the Details tab.
3. Click the New icon on the task bar.

   This action causes a blank entry to display in the Services list at the left.
4. Enter a value for the Display Name.

   This is the name of the service that is to display in the audit trail and during configuration.
5. Enter a value for the Internal Name.

   This is the name for the service that is to be used internally by the system during provisioning.
6. Enter a version number.
7. Check the Provision this Service box.
   - If you check this box, then you can submit an order against the service.
   - If you do not check this box, then the configuration of this service is still in progress, and an order can not be submitted against the service.
8. Enter a short description in the Description field.
9. Enter the effective date for this service to become active.
10. Enter the effective date for this service to become inactive.
11. Enter a value for Responsibility.

    This sets who is responsible to maintain this information.
12. Close the window.
13. Save your changes.

To complete the definition, you need to associate an action with this service.

## Associating Actions with a Service

An action is an operation performed on a service, product, and package. It instructs the Service Delivery Platform to internally call pre-defined sets of activities, rules, or data.

For every service that you define, you need to assign an action for provisioning that desired service. A service combined with an action creates the definition of an order line item.

You can repeat an action across various services, however, this action may not necessarily be valid across all fulfillment elements.

To create a new service action, perform the following steps.

---

**Note:** It is possible to associate multiple actions with a single service.

---

### Prerequisites

You must define a service before associating actions with it. (You may also reuse an existing service.) See Defining a New Service for details.

### Steps

1. In the Navigator, choose **Setup > Service Definitions > Services**.
2. Select the Actions tab.
3. Select an action from the Available Actions list.
   - If you do not see the name of the desired action in the Available box, enter the name of the action in the Filter field and click the **Find** icon to search for it.
   - If the action is not defined yet, click **Define Actions** to open the Define Actions window and then add it.
4. Use the arrow buttons to move your choice from the Available Actions list to the Selected Actions list.
5. Enter a value for the Start Date.

   This is the date at which this action becomes available to the service.
6. Enter a value for the End Date.

   This is the date at which this action becomes unavailable.
7. Enter a Description for this action.

8. Enter a name for the Work Item Mapping Procedure.

   If necessary, click the Add/Edit Procedure icon to launch the procedure builder and define a new procedure. See Using the Procedure Builder for details of this process.

9. Close the window.

10. Save your changes.

### Associating Work Items with Service Actions

To associate a work item with a service action, perform the following steps.

### Prerequisites

Each service action can have one or more work items defined for it. However, there must be at least one work item for any available service action.

### Steps

1. In the Navigator, choose **Setup > Service Definitions > Services**.

2. Select a service from the list at the left.

3. Select the Work Items for Action tab.

4. Select an action from the drop down list of available actions associated with this service.

   If you do not see the action that you desire, you need to select the Actions tab and add the desired action to the list of Selected Actions.

5. Select a work item from the Available Work Items list.

   - If you do not see the name of the desired work item in the Available box, enter the name of the work item in the Filter field and click the Find icon to search for it.
   - If the work item is not defined, click **Define Work Items** to open the Define Work Items window and then add it. See Defining a New Work Item for details, if necessary.

6. Use the arrow buttons to move your choice from the Available Work Items list to the Selected Work Items list.

7. Enter a Provisioning Sequence numb er for each item in the Selected Work Items list.
   - Fulfillment actions marked with a 1 have no dependencies on other fulfillment actions and are processed in parallel with all other fulfillment actions marked 1.
   - Fulfillment actions marked with a 2 have a dependency on another fulfillment action, and are processed after all fulfillment actions marked with a 1 are processed.
8. Enter a descriptive comment in the Details of a Selected Work Item field, if desired.
9. Repeat steps 4 through 8 as many times as necessary.
10. Close the window.
11. Save and exit.

### Editing an Existing Action

To edit an existing service action, perform the following steps.

### Prerequisites

None

### Steps

1. In the Navigator, choose **Setup > Service Definitions > Services**.
2. Select the Action tab.
3. Click **Define Actions**.
4. Select an existing action from list.
5. Modify the action parameters as desired.
6. Close the window.
7. Save your changes.

### Deleting a Service

To delete a service, perform the following steps.

#### Prerequisites

You can not delete a service if either the following conditions is true:

- If the allowable actions on the service have previously been defined.
- If orders have been running for that service.

#### Steps

1. In the Navigator, choose **Setup > Service Definitions > Services**.
2. Select a service from the list at the left.
3. Click **Delete**.

   The service disappears from the list.
4. Close the window.
5. Save your changes.

## Packages

You perform several different tasks with service packages. These include:

- Defining a new service package
- Associating services with packages

### Defining a New Service Package

To create a new package of services, perform the following steps.

#### Prerequisites

None

### Steps

1. In the Navigator, choose **Setup > Service Definitions > Packages**.
2. Select the Details tab.
3. Enter a value for the Display name.

   This is the name of the package that is to display in the audit trail and during configuration.
4. Enter a value for the Internal name.

   The system stores this name in its database to identify the package. (Use underscores rather than blanks in the package name.)
5. Enter a value for the Description.
6. Enter a value for the Start Date.

   This is the date at which this package becomes available.
7. Enter a value for the End Date.

   This is the date at which this package becomes unavailable.
8. Close the window.
9. Save your changes.

## Associating Services with Packages

To associate a service with a package, perform the following steps.

### Prerequisites

You must define the service package before you add services to it. See Defining a New Service Package for details.

### Steps

1. In the Navigator, choose **Setup > Service Definitions > Services**.
2. Select the Services tab.
3. Select a service from the Available Services list.
   - If you do not see the name of the desired service in the Available box, enter its name in the Filter field and click the Find icon to search for it.
   - If the service is not defined yet, click **Define Services** and then add it. See Defining a New Service for details.
4. Use the arrow buttons to move your choice from the Available list to the Selected list.
5. Enter an Activation Sequence value for each service listed under Selected Services.

   This value determines the order in which the listed services are activated.
6. Check Provisioning Required if this service is to be provisioned by the Service Delivery Platform.
   - If you check this box, then the service is provisioned through the Service Delivery Platform
   - If you do not check this box, then the service is defined within the Service Delivery Platform Configurator, but is not necessarily activated through the Service Delivery Platform.

   This feature enables Service Providers to implement the identical service name among different provisioning medium without conflict.
7. Enter a comment in the Comment field, if desired.
8. Close the window.
9. Save your changes.

## Work Items

You perform a number of tasks with work items. These include:

- Defining a new work item
- Deleting a work item
- Defining a new work item parameter
- Adding parameters to a work item
- Associating fulfillment actions with work items

### Defining a New Work Item

To define a new work item, perform the following steps.

### Prerequisites

If you plan to associate a workflow with this work item, then you must create the workflow *before* you create the work item.

### Steps

1. In the Navigator, choose **Setup > Service Definitions > Work Items**.
2. Click the New icon on the toolbar.

   This action causes a blank Details window to display.
3. Enter the Display name.

   System screens display this name for the work item.
4. Enter an Internal name.

   The system stores this name for the work item in its database.
5. Enter the Version.

   The Service Delivery Platform allows multiple versions of a work item.
6. Enter the Duration.

   The work item stays active for this length of time.
7. Enter a Description for this work item.
8. Enter the Begin Date.

   The work item becomes available on this date.

9. Enter the End Date.

   The work item is no longer available after this date.

10. Chose a Work Item Type from the drop down list.

    See Guidelines following for details of work item types.

11. Chose a user role from the drop down list for Responsibility.

    This determines who is responsible to maintain this information (a user, a group, or a user group). It can also be used for fault resolution assignment (i.e., assigning it to the person responsible for ensuring that the problem is resolved).

12. Close the window.

    You are prompted to save your changes.

### Guidelines

Use the following information in determining the type of work item to choose.

***Work Item Types***

| WI Type | Comments |
|---|---|
| Static | If selected, then you must also define the fulfillment actions associated with this work item.<br>See Associating Fulfillment Actions with Work Items for details. |
| Dynamic | If selected, then you must also select a procedure from a list of existing procedures or create a new procedure. The procedure determines the list of fulfillment action's and their sequence at runtime. |
| User-defined workflow | If selected, enter the Item Type, Process and Key Prefix.<br>You must also enter the name of the workflow that you have created. (Remember that any workflow that you create must be saved to the general Service Delivery Platform engine before it is accessible for use.) |
| User workflow start procedure | If selected, then you must enter the name of the workflow procedure also, or define a new one. |

### Deleting a Work Item

To delete an existing work item, perform the following steps.

### Prerequisites

The data must be visible in your log in role for you to delete it.

### Steps

1. In the Navigator, choose **Setup > Service Definitions > Work Items**.
2. Select a work item from the list at left.
3. Click **Delete.**

   The system performs a validation procedure to determine that there are no ill effects to deleting this work item. For example, it checks to see that no parameters or fulfillment actions are assigned to the work item to be deleted. If if passes the validation procedure, then the work item is deleted by the system and it is removed from the list.

## Defining a New Work Item Parameter

To define a new work item, perform the following steps.

### Prerequisites

None

### Steps

1. In the Navigator, choose **Setup > Service Definitions > Work Items**.
2. Select the Parameters tab.
3. Click **Define Parameters**.

   The Define Parameters window opens.
4. Click the New icon on the toolbar.

   A blank entry opens in the Display Name list at the left.
5. Enter a value for the Internal Name of the parameter.

   The Service Delivery Platform stores this name in its database fro internal use by the system.
6. Enter a description for the new parameter.
7. Close the window.
8. Save your work.

### Adding Parameters to a Work Item

To add parameters to a work item, perform the following steps.

### Prerequisites

None

### Steps

1. In the Navigator, choose **Setup > Service Definitions > Work Items**.
2. Select the Parameters tab.
3. Select a parameter from the Available Parameters list.

   If you do not see the name of the desired parameter in the Available box, enter its name in the Filter field and click the **Find** icon to search for it.
4. Use the arrow buttons to move your choice from the Available list to the Selected list.
5. (Optional) Enter a Display Sequence number for each item in the Selected Parameter list.

   This value determines the order in which the parameter displays in the work item sequence.
6. (Optional) Enter a Evaluation Sequence number for each item in the Selected Parameter list.

   This value determines the order in which the work item evaluates the parameter.
7. Check the Required box if this parameter must be present in order to provision the service.
8. Check the Log in Audit Trail box if this parameter is to be logged and visible in the audit trail.

   Parameters that have this box checked (and are actually used with an order), become visible in the audit trail.
9. Enter a Default Value, if desired.

   This value is displayed in the order, and is changed by the user. For example, this could be an IP address or a password.

10. Enter a value in the Value Lookup SQL field.

    This value is an SQL statement that evaluates the parameter value. If necessary, click **Edit Lookup SQL** to launch the SQL editor.

11. Enter a mode for when evaluation of this parameter occurs, if desired.

    The available choices are:

    - Upon order receipt
    - When provisioning order

    See the Work Item Mode Guidelines following for considerations on choosing a work item mode.

12. (Mandatory, if a mode is selected in step 11, otherwise, ignore.) Enter a name for the procedure to be used in evaluating the parameter value.

    If necessary, click the Add/Edit procedure icon to launch the procedure builder and define a new procedure. See Using the Procedure Builder for details of this process.

13. Close the window.

14. Save your work.

### Work Item Mode Guidelines

You may wish to review the following information before selecting a work item mode.

### Order Receipt

If you select **Upon Order Receipt**, then you must also define a name for the procedure and write the body for the evaluation procedure.

The following steps are performed by the Service Delivery Platform if a parameter is marked as Upon Order Receipt. The Service Delivery Platform:

- Receives an order from the upstream system with the parameter and its value.
- Performs validation on the parameters.
- Executes the evaluation procedure.
- Stores the value of the parameter returned from the evaluation procedure. (This value is used in further processing.)

### Order Provisioning

If you select Upon Provisioning Order, then you must also define a name for the procedure and write the body for this procedure.

The following steps are performed by the Service Delivery Platform if a parameter is marked as When Provisioning Order. The Service Delivery Platform:

- Receives an order from an upstream system with the parameter and its value.
- Performs validation on the parameters.
- Stores the value of the parameter as is, without any further manipulation.
- At runtime, whenever the parameter is referenced, the associated When Provisioning Order procedure is executed to obtain the new value.
- The new value is used for order processing.

### Associating Fulfillment Actions with Work Items

To associate a fulfillment action with a work item, perform the following steps.

### Prerequisites

You can only associate fulfillment actions with either static or dynamic work items.

### Steps

1. In the Navigator, choose **Setup > Service Definitions > Work Items**.
2. Select a work item from the list at left.
3. Select the Fulfillment Actions tab.

   The work item type must be static for the Fulfillment Actions tab to be accessible.
4. Select a fulfillment action from the Available Fulfillment Actions list.
   - If you do not see the name of the desired fulfillment action in the Available box, enter the name of the fulfillment action in the Filter field and click the **Find** icon to search for it.
   - If the user account is not defined yet, click **Define Fulfillment Actions** to open the Define Fulfillment Actions window and then add it. See Configuring Fulfillment Actions for details on adding a new fulfillment action.
5. Use the arrow buttons to move your choice from the Available Fulfillment Actions list to the Selected Fulfillment Actions list.

6. Enter a Provisioning Sequence number for each item in the Selected Fulfillment Actions list.
   - Fulfillment actions marked with a 1 have no dependencies on other fulfillment actions and are processed in parallel with all other fulfillment actions marked 1.
   - Fulfillment actions marked with a 2 have a dependency on another fulfillment action, and are processed after all fulfillment actions marked with a 1 are processed.
7. Enter a Comment for this fulfillment action, if desired.
8. Close the window.
9. Save your changes.

## Managing Fulfillment Actions

You perform a number of tasks with fulfillment actions. These include:

- Configuring fulfillment actions
- Adding parameters to a fulfillment action
- Associating a fulfillment procedure with a fulfillment action
- Modifying a fulfillment action procedure

### Configuring Fulfillment Actions

To define a new fulfillment action, perform the following steps.

### Prerequisites

You must perform the following tasks before configuring fulfillment actions:

- **Configure parameters:** See Adding Parameters to a Work Item.
- Configure fulfillment element types: See Defining a Fulfillment Element Type.
- **Configure fulfillment elements:** See Defining a New Fulfillment Element.
- **Build the fulfillment procedures:** See Using the Procedure Builder.

### Steps

1. In the Navigator, choose **Setup > Service Definitions > Fulfillment Actions**.

   The Fulfillment Actions window opens.

2. Click the New icon on the toolbar, or chose **File > New** from the menu.

   A new, blank entry opens in the Fulfillment Actions list at the left.

3. Enter a value for the Display name.

   This is the name of the fulfillment action that is to display in the audit trail and during configuration.

4. Enter a value for the Internal name.

   This is the name for the fulfillment action that is to be used internally by the system during provisioning.

5. Enter the Version number for this fulfillment action.

6. Enter a value for the Description.

7. Chose one of the available Fulfillment Element Routing procedures from the drop down list.

   If necessary, click the icon next to the Procedure field and create a routing procedure for use here. See Using the Procedure Builder for details.

8. Enter a value for Evaluate a Parameter.

   If necessary, click the icon next to the Procedure field and create an evaluation procedure for use here. See Using the Procedure Builder for details.

9. Chose a Responsibility from the drop down list, if desired.

   If this field is left blank, any user role can use this fulfillment action.

10. Save your work.

### Adding Parameters to a Fulfillment Action

To associate parameters with a fulfillment action, perform the following steps.

**Note:** A fulfillment action can have more than one parameter associated with it.

### Prerequisites

None

### Steps

1. In the Navigator, choose **Setup > Service Definitions > Fulfillment Actions**.
2. Select a fulfillment action from the list at the left.
3. Select the Parameters tab.
4. Select a parameter from the Available Parameters list.
   - If you do not see the name of the desired parameter in the Available box, enter its name in the Filter field and click the **Find** icon to search for it.
   - If the parameter is not defined yet, click **Define Parameters** and then add it. See Defining a New Work Item Parameter for details.
5. Use the arrow buttons to move your choice from the Available list to the Selected list.
6. (Optional) Enter a Display Sequence number for each item in the Selected Parameter list.

   This value determines the order in which the parameter displays in the fulfillment action sequence.
7. Check the Log in Audit Trail box if this parameter is to be logged and visible in the audit trail.
8. Enter a Default Value, if desired.

   This value is displayed in the order, and is changed by the user. For example, this could be an IP address or a password.
9. Enter a name for the Evaluation Procedure to be used in evaluating the parameter value. Specify an evaluation procedure for each parameter that you need to define, if desired.

   If necessary, click the icon next to the Procedure field to launch the procedure builder and define a new procedure. See Using the Procedure Builder for details of this process.

10. Click the Save icon on the toolbar to save your changes.
11. Select **File > Close** to exit.

### Associating a Fulfillment Procedure with a Fulfillment Action

To associate fulfillment procedures with a fulfillment action, perform the following steps.

#### Prerequisites

A single fulfillment action can have more than one fulfillment procedures defined. However, it must have at least one procedure to indicate the adapter type.

#### Steps

1. In the Navigator, choose **Setup > Service Definitions > Fulfillment Actions**.
2. Select a fulfillment action from the list at left.
3. Select the Fulfillment Procedures tab.
4. Chose a Fulfillment Element Type from the drop down list.

   If necessary, click **Define Types** and create a new fulfillment element type.
5. Enter a value for Software Version (Adapter Type) to set the software version and the connect / disconnect values.
6. Enter a value for the fulfillment procedure to select the PL / SQL procedure to execute.

   If necessary, click the icon next to the Procedure field and create a new procedure for use here.
7. Click the Save icon on the toolbar to save your changes.
8. Select **File > Close** to exit.

### Modifying a Fulfillment Action Procedure

To modify or edit an existing fulfillment procedure, perform the following steps.

#### Prerequisites

None

#### Steps

1. In the Navigator, choose **Setup > Service Definitions > Fulfillment Actions**.
1. Click the icon next to the Procedure field that you wish to modify.

   The Add/Edit Procedure window opens.
2. Modify the procedure as desired.
3. Click the Save icon on the toolbar to save your changes.
4. Select **File > Close** to exit.

## Procedure Builder

You use the procedure builder to perform the following tasks:

- Creating a new procedure
- Deleting an existing procedure
- Viewing procedure parameters
- Editing an existing procedure

### Creating a New Procedure

To create a new procedure, perform the following steps.

#### Prerequisites

None

### Steps

1. Open the Add/Edit Procedure window by clicking the icon next to the Procedure field.
2. Verify the procedure Type.

   This field is read-only. Its value depends on the form field that called the Add/Edit Procedure window.
3. Enter the internal name for this procedure.

   The Service Delivery Platform uses this name to reference the procedure in its database.
4. Enter the display name.

   This name is displayed in the system screens.
5. Enter a description of this procedure.
6. Do one of the following:
   - Enter the desired PL/SQL code in the Procedure code field.
   - Click **Copy From** to copy the code from an existing procedure. Use the Edit function to modify it to meet your needs.

   See Guidelines following for details on guidelines to use during the creation of a procedure.
7. Click **OK** to compile and generate the procedure.

### Guidelines

Use the following guidelines during creation of a procedure.

- Do **not** use an initial DECLARE statement at the beginning of the code block
- Use the provided *Script Notepad* to write your provisioning procedure.
- Write your block of code as an anonymous PL/SQL block. (The following is an example of a PL/SQL connection procedure.):

```
  err_code number;
  err_str     varchar2(400);
BEGIN
  LOGIN('telnet $ip_address $portnumber', '$username_prompt', err_code,
err_str);
     SEND('$username', '$password_prompt', err_code, err_str);
     SEND('$password', '$login_prompt', err_code, err_str);
```

```
EXCEPTION
WHEN OTHERS THEN
   RAISE;
END;
```

- Use capital letters for all PL/SQL reserved words and the constructs provided by the Service Delivery Platform.
- Use the other standard programming constructs provided by PL/SQL in your block of code as desired.
- Always add the EXCEPTION block at the end of your procedure:

```
EXCEPTION
WHEN OTHERS THEN
   RAISE
END;
```

- You must compile and generate the procedure after finishing the code by clicking OK.

  The procedure builder prompts you with compilation errors if you have either of the following conditions exist:

  - The procedure contains PL/SQL syntax errors.
  - The procedure contains unrecognized parameters.

  **Note:** To display a list of the parameters that can be used in the provisioning procedure, click **Available Parameters**.

  The displayed parameters are the internal names of the service parameters that you previously configured. Only these internal names can be used inside the provisioning procedure body.

### Deleting an Existing Procedure

To delete an existing procedure, perform the following steps.

### Prerequisites

The procedure to be deleted must **not** be used by any other objects.

### Steps

1. Open the Add/Edit Procedure window by clicking the icon next to the Procedure field.
2. Verify that the procedure Type lists the procedure that you wish to delete.
3. Click the Delete icon on the toolbar.
4. Click the Save icon on the toolbar to save your changes.
5. Select **File > Close** to exit.

### Viewing Procedure Parameters

To view the parameters or attributes which can be used in the procedure, perform the following steps.

### Prerequisites

None

### Steps

1. Open the Add/Edit Procedure window by clicking the icon next to the Procedure field.
2. Click **Attributes**.
3. Click **OK** to close the Parameters window.
4. Click **OK** to exit the Add/Edit Procedure window.

---

**Note:** The parameters or attributes displayed are the internal names of the fulfillment element attributes that you have previously configured. Only internal names can be used inside the procedure body.

---

### Editing an Existing Procedure

To edit an existing procedure, perform the following steps.

### Prerequisites

Do not modify a default procedure.

### Steps

1. Open the Add/Edit Procedu**re** window by clicking the icon next to the Procedure field.
2. Verify the procedure type.

   This field is read-only. Its value depends on the form field that called the Add/Edit Procedure window.
3. Click on the Procedure Code field.
4. Select **Edit > Edit Field** from the menu.
5. Modify the PL/SQL code in the Procedure Code text field.
6. Click **OK** to compile and generate the procedure.

## Managing Fulfillment Element Types

For every fulfillment element that you define, there must first exist a definition of that fulfillment element type.

The following tasks are associated with fulfillment elements types:

- Defining a fulfillment element type
- Deleting a fulfillment element type

### Defining a Fulfillment Element Type

> **Note:** Any service order request provisioned through the Service Delivery Platform requires that a specific fulfillment order type for that request also be configured through the Service Delivery Platform.

To define a new fulfillment element type, perform the following steps.

### Prerequisites

Fulfillment element types are defined by system experts only.

### Steps

1. In the Navigator, choose **Setup > Interface Definition > Fulfillment Element Types**.

   The Fulfillment Element Types window opens.

2. Select the Details tab.
3. Click **Define Types**.
4. Click **Add**.

   This action causes a blank Define Fulfillment Element Types window to open.

5. Complete the fields in the Details tab.
6. Complete the fields in the Attributes tab.
7. Complete the fields in the Software tab.
8. Close the window.
9. Save your changes.

## Deleting a Fulfillment Element Type

To delete an existing fulfillment element type, perform the following steps.

### Prerequisites

You can not delete a fulfillment element type if either one of the following conditions is true:

- The attributes for this fulfillment element have previously been defined
- The software versions for this fulfillment element previously have been defined

### Steps

1. In the Navigator, choose **Setup > Interface Definition > Fulfillment Element Types**.

   The Fulfillment Element Types window opens.

2. Select the Details tab.
3. Select the fulfillment element type to delete from the list at the left.
4. Click Delete icon on the toolbar.
5. Close the window.
6. Save your changes.

## Managing Fulfillment Elements

There are a number of tasks associated with fulfillment elements. These include:

- Defining a new fulfillment element
- Deleting a fulfillment element
- Modifying a fulfillment element configuration

### Defining a New Fulfillment Element

To add a new fulfillment element, perform the following tasks.

> **Warning: Each Fulfillment Element name must be unique and consist only of alphanumeric characters.**

### Prerequisites

The type of fulfillment element that you wish to add must exist before you can add the fulfillment element.

### Steps

1. In the Navigator, choose **Setup > Interface Definition > Fulfillment Elements**.
2. Select the Details tab.
3. Click the New icon on the toolbar.

   This action causes a blank Details screen to open.
4. Complete the fields in the Details tab.
5. Complete the fields in the SW Versions tab, including both the Details and Attributes sub-tabs.
6. Close the window.
7. Save your changes.

## Deleting a Fulfillment Element

To delete a fulfillment element, perform the following tasks.

> **Warning: If you delete a fulfillment element, then the logical and physical entity of the fulfillment element is removed from the Service Delivery Platform database.**

### Prerequisites

You can *not* delete a fulfillment element, if either the following is true:

- You have previously defined attributes for this fulfillment element.
- You have previously defined the software generics for this fulfillment element.

### Steps

1. In the Navigator, choose **Setup > Interface Definition > Fulfillment Elements**.
2. Select the Details tab.
3. Select the fulfillment element that you wish to delete from the list at the left.
4. Click **Delete**.
5. Close the window.
6. Save your changes.

### Modifying a Fulfillment Element Configuration

If changing a fulfillment element configuration (for example, disabling a fulfillment element, or changing the fulfillment element attributes such as the IP address), perform the following steps.

#### Prerequisites

None.

#### Steps

1. Shutdown the Work Item queue to stop the work item dequeuers.
2. Ensure that there are no fulfillment actions remaining in the Wait for FE queue and the FE Ready queue. This means that all the fulfillment actions are processed at this point.
3. Shutdown the adapters for the fulfillment element.
4. Change the attributes for the fulfillment element.
5. Re-start the adapters for the fulfillment element.
6. Re-start the Work Item queue.

### Creating an Activity or Process Timer

Perform the following steps to create either an activity or process timer.

#### Prerequisites

None

### Steps

1. In the Navigator, choose **Setup > Message Definition > iMessage Studio**.
2. Click the New icon on the toolbar to create a new timer message.

   A blank field opens in the Message Code list.
3. Enter a name for the new timer in the Message Code field.

   The system uses this name for internal identification.
4. Select the Details tab.

   The Elements and Structure tabs are grayed out. Timer messages come with two pre-defined, mandatory elements: Interval and Duration.
5. Select Timer from the drop-down list of available types.
6. Enter a short name for the timer in the Display Name field.
7. Enter a brief description for the timer.
8. Select a priority for the timer.

   This value sets the priority for the timer message in the Timer Message queue.
9. Select Timer Queue from the Queue Name drop-down list of values.
10. (Optional) Choose a responsibility for this message from the drop-down list of values.
11. Click the Save icon on the toolbar.

    You must save your work before proceeding.
12. Select the Data Source tab.
13. Select the tree root for the timer.
14. Select SQL Query from the Data Source Type drop-down list of values.
15. Select One and Only One from the Data Source Cardinality drop-down list of values.
16. Specify the source code for the timer in the Source field.

    See the Source Guidelines following for details.
17. Select the Delay branch from the tree root for the activity timer.
18. Enter a reference for the Delay branch in the Reference field.

    See the Delay Interval Guidelines following for details.

19. Select the Interval branch from tree root for the Activity Timer.
20. Enter a reference for the Interval branch in the Reference field.

    See the Interval Guidelines following for details.
21. Click the Save icon on the toolbar.
22. Select the Detail tab.
23. Click **Compile** to compile the timer.

    If an error occurs, verify that you have followed steps 2 - 23 above correctly. After correcting the error, recompile the message.
24. Click the Close Form icon on the toolbar to close this form window.

### Source Guidelines

The source SQL structure for the activity timer must be of the following format:

```
SELECT '<D>' DELAY, '<I>' INTERVAL FROM DUAL
```

Where

- <D> is the delay integer value (in seconds)
- <I> is the interval integer value (in seconds)

For example:

```
SELECT '0' DELAY, '900' INTERVAL FROM DUAL
```

### Delay Guidelines

The reference value must be of the following format:

```
xnp$<activity_timer_name>.<select-list column name>
```

Where

- <activity_timer_name> is the internal name defined in step 3 above.
- <select-list column name> is the first select-list value of the SQL source specified in step 16.

For example:

```
XNP$ACT_TIMER_1.DELAY
```

### Interval Guidelines

The reference value must be of the following format:

```
xnp$<activity_timer_name>.<select-list column name>
```

Where

- <activity_timer_name> is the Internal Name defined in step 3 above.
- <select-list column name> is the second select-list value of the SQL source specified in step 16.

For example:

```
XNP$ACT_TIMER_1.INTERVAL
```

### Creating a Message Timer

Perform the following steps to create a message timer.

### Prerequisites

Before creating the message timer, you must first create the message with which it is associated.

### Steps

1. In the Navigator, choose **Setup > Message Definition > iMessage Studio**.
2. Click the New icon on the toolbar to create a new timer message.

   A blank field opens in the Message Code list.
3. Enter a name for the new timer in the Message Code field.

   The system uses this name for internal identification.
4. Select the Details tab.
5. Select Message from the drop-down list of available types.
6. Enter a short name for the timer in the Display Name field.
7. Enter a brief description for the timer.
8. Select a priority for the timer.

   This value sets the priority for the timer message in the Timer Message queue.

9. Select Outbound Message Queue from the Queue Name drop-down list of values.
10. (Optional) Choose a responsibility for this message from the drop-down list of values.
11. (Optional) Enter a path reference in the DTD Location field.
12. Click the Save icon on the toolbar.

    You must save your work before proceeding.
13. Use the Elements tab to add elements to the message.
14. Use the Structure tab to define the structure of the message.
15. Save your work before closing the window.

### Workflow Procedure Guidelines

Use the following rules during procedure definition.

- Use the provided **procedure notepad** to write your provisioning procedure.
- Write an anonymous PL/SQL block as to insert into the procedure body.
- Use capital letters for all PL/SQL reserved words and the constructs provided by Oracle Product Name.
- Use the standard programming constructs provided by PL/SQL in your block of code, if desired.
- Always add the **EXCEPTION** block at the end of the procedure.

### Example Procedure Definition

The procedure builder provides a structure for building your procedures. For example, to build a new procedure, you simply insert your code into the following procedure definition.

```
/*****************************************************************************
This procedure calculate a new value for the Work Item parameter.
It has the follwoing input and output parameters:
p_order_id           IN NUMBER   -- order ID
p_wi_instance_id     IN NUMBER   -- wprlote, omstamce OD
p_param_val          IN VARCHAR2 -- parameter initial value
p_param_ref_val      IN VARCHAR2 -- reference value (if order amendment)

p_parmam_eval_val    OUT VARCHAR2 -- parameter new value
p_param_eval_val     OUT VARCHAR2 -- parameter new value
p_param_eval_ref_val OUT VARCHAR2 -- new reference value (if order ammendment)
*****************************************************************************/
```

### Procedure

Enter your procedure below:

```
DECLARE

-- Your declarations...

BEGIN
-- Your procedure body...
END;

:
EXCEPTION
WHEN OTHERS THEN
RAISE
END;
```

### Compiling Procedures

After creating a new procedure, you must compile it. The procedure builder prompts you with compilation errors if you have either of the following conditions exist:

- The procedure contains PL/SQL syntax errors.
- The procedure contains unrecognized parameters.

### The AOL Generic Loader

The Oracle Application Object Library loader is a general purpose data migration tool that is used for patching seed data, delivering translations, or copying setup or transaction data from development to production systems.

The loader is a concurrent program named FNDLOAD. To use this utility, enter the following command at a UNIX prompt.

```
FNDLOAD apps/pwd 0 Y mode configfile datafile entity [ param ... ]
```

The table following lists the parameters used with this executable and describes them.

| Parameter | Description |
|---|---|
| apps/pwd | Specifies the APPS schema and password.<br>▫ If the connect_string is omitted, it is taken in a platform-specific manner from the environment using the name TWO_TASK |
| 0 Y | Concurrent program flags |
| mode | Specifies either UPLOAD or DOWNLOAD operation.<br>▫ UPLOAD causes the specified data file to be uploaded to the database.<br>▫ DOWNLOAD causes the loader to fetch rows and write them to the specified data file. |
| configfile | Specifies the configuration file to use.<br>▫ The configuration file usually ends with a suffix of.lct, but this rule is neither enforced nor supplied by the loader. |
| datafile | Specifies the data file to write.<br>▫ (DOWNLOAD) If the data file already exists, then it is overwritten.<br>▫ The configuration file usually ends with a suffix of.lct, but this rule is neither enforced nor supplied by the loader. |
| entity | Specifies the entity type to begin the download or upload.<br>▫ If you wish to upload all of the entity types in a data file (.ldt), specify a dash (-) as the entity type. |
| param | Specifies zero or more additional parameters that are used to provide bind values in the access SQL (for both the UPLOAD and DOWNLOAD operations).<br>▫ Each parameter is of the form NAME=VALUE. The given NAME must not conflict with an attribute name for the entities being loaded. |

### Loader File Definitions

You can find the FNDLOAD configuration files for XDP at the following location:

```
$XDP_TOP/patch/115/import/*.lct
```

The table following lists the loader files used with Oracle Provisioning and provides the entities and download parameters supported by each. See the contents of the individual configuration file for full documentation on usage.

***Loader File Description (Optional Download Parameters)***

| Name | Description | Entity | Optional Download Parameters |
|---|---|---|---|
| xdpparpl.lct | Parameter Pool | XDP_PARAMETER_POOL | PARAMETER_NAME |
| xdpprbod.lct | Procedure Body | XDP_PROC_BODY | PROC_NAME |
| xdpaccod.lct | Action Codes | XDP_ACTION_CODES | ACTION_CODE |
| xdpfacts.lct | Fulfillment Actions | XDP_FULFILL_ACTIONS | FULFILLMENT_ACTION |
| | | | VERSION |
| xdpwipar.lct | Work Items | XDP_WORKITEMS | WORKITEM_NAME |
| | | | VERSION |
| xdpfetyp.lct | Fulfillment Element Types | XDP_FE_TYPES | FULFILLMENT_ELEMENT_TYPE |
| xdpfes.lct | Fulfillment Elements | XDP_FES | FULFILLMENT_ELEMENT_NAME |
| xdpsrves.lct | Services | XDP_SERVICES | SERVICE_NAME |
| | | | VERSION |
| xdppkges.lct | Packages | XDP_SERVICE_PACKAGES | PACKAGE_NAME |
| | | | PACKAGE_VERSION |

The following tables lists dependencies between the various files.

| Name | Dependency |
|---|---|
| xdpfacts.lct | First run xdpparpl.lct |
| xdpwipar.lct | First run xdpparpl.lct |
| xdpfes.lct | First run xdpfetyp.lct |
| xdpsrves.lct | First run xdpaccod.lct. |
| xdppkges.lct | First run xdpsrves.lct. |

Several of the loader files also support optional upload parameters. The table following lists them.

***Loader File Description (Optional Upload Parameters)***

| Name | Upload Parameter | Comments |
|---|---|---|
| xdpwipar.lct | OVERRIDE_MAPPNG | ▫ If this command line argument is set to 'Y', then all existing mappings from a work item to a fulfillment action are deleted first. Then, the mappings being inserted by the loader.<br>▫ If this command line argument is set 'N' or if it is not defined, mapping are inserted for a work item only if it did not exist previously. |
| xdpsrves.lct | OVERRIDE_MAPPING | ▫ If this command line argument is set to 'Y', then all existing mapping from service valid actions to work items are deleted first. Then the mappings are inserted by the loader.<br>▫ If this command line argument is set 'N' or if it is not defined, mapping are inserted for service valid actions only if it did not exist previously. |

### References

For additional information, see the files in the following application directories.

- **Template configuration file**

  /fnddev/fnd/11.5/admin/import/fndstd.lct

- **Existing AOL configuration files**

  /fnddev/fnd/11.5/admin/import/*.lct

## Event Subscription

Applications register with the Event Manager if they need information on specific messages (events) or all instances of that message type. An application can be an external system, an operating system or a mediation layer.

Every message and event is sent to the Event Manager which becomes the source of distribution for events and messages. This is known as event subscription. Event subscription is a business process handled within Workflow.

- Applications can be set as default subscribers to specific events and messages. This is configured in the Event Subscriber utility.
- Applications can show interest in a specific instance of a message by specifying the REFERENCE_ID of the message.
- Applications can register for one of several response messages by using the reference identifier.
- When a message with a reference identifier arrives, the Event Manager performs the following tasks:
  - It executes the validation logic.
  - It executes the processing logic. It does this before delivering the message to all registered applications.
- Applications can show interest in all instances by registering with a NULL reference identifier.
- If an application is registered for a specific instance, the registration is automatically unsubscribed after delivering that message.
- Upon arrival of an expected response:
  - The message is delivered to the registered application.
  - The other expected response is marked EXPIRED.

### Setting Up Event Subscription

In the Event Manager, it is possible for default subscribers to subscribe to specific events of the application. When a message occurs, the Event Manager ensures that an outbound message is automatically sent to the subscriber identified by a Fulfillment Element, OSS or gateway.

### Responding to an Event

- It is possible to associate one or more responses to an event.
- A response is set during the configuration in the form of acknowledgments.

### Steps

1. Create the message or event that is to the trigger for a response.
2. Configure an Acknowledgment / Response to the message or event that you created in step 1.
3. Link the Acknowledgment that you created in step 2 to the message / event that you created in step 2.

### Automatic Responses

If desired, you can associate one or more responses with an event. A response is an acknowledgment to a message.

For example, valid message responses for the message "Is this an existing customer?" are:

- "Yes, this is an existing customer."
- "No, this is not an existing customer."

These responses are messages in themselves and must be configured in Oracle Provisioning Name before they can be linked as responses to a message.

## Administering the Oracle Service Delivery Platform

The Concurrent Manager framework provided by Oracle Applications is used to administer the Oracle Provisioning Concurrent Manager. The Concurrent Manager includes a specialized Controller program that is used by the Oracle Service Delivery Platform to start (or stop) adapters and / or dequeuers.

Orders within the Service Delivery Platform are processed using various queues. The processing of items of a queue is done with the help of **dequeuers**. These dequeuers are background processes which continuously poll a queue and process the items from the queue.

The provisioning of various network elements is accomplished with the use of adapters. Adapters are processes which are specialized in certain protocols. (Telnet, FTP, or other protocols.). You direct the adapters to perform various tasks in the system.

**Note:** To administer the Oracle Provisioning Concurrent Manager, open the Administer Concurrent Managers form window available through **Concurrent > Administer Manager**.

### Oracle SDP Start

Application Start is a concurrent request that is performed by the Oracle Provisioning Concurrent Manager.

You use the Application Start process to perform the following tasks:

- Start the entire application, including the Controller, dequeuers and all adapters
- Start only the Controller and the dequeuers
- Start the Controller, the dequeuers and a subset of the adapters

The system automatically starts all the dequeuers and the Controller during the start process. If any dequeuers are already started, the number actually running is first determined, and matched against the number that are required to be started. The system automatically starts as many dequeuers as needed.

**Note:** You set the number of running dequeuers through the Queue Console, available through **Administration > Queue Console**.

### Prerequisites

Application Start and Stop are mutually exclusive processes. If one is pending, the opposite operation does not proceed until the first finishes. In addition, the Oracle Provisioning Concurrent Manager processes must be up and running.

### Steps

1. In the a Navigator, select **Applications > Start**.

   The Parameters box opens.

2. Enter a value in the Options field from the drop-down list of values.

   See the Guidelines following for information on these choices.

3. Enter a value in the Fulfillment Element field that is to be included (or excluded) from the application start process.

   This choice depends on the value that you chose in step 2, preceding. Again, see the Guidelines for details.

4. Enter a value (0 - 3) for the Debug Mode.

   This value sets the level of logging that can be used for debugging purposes.

5. Click **OK**.

   The Oracle SDP Start dialog box becomes available. Parameters that you set in the previous dialog box display in the Parameters field.

6. (Optional) Click **Copy**, if you wish to use start parameters from a prior start request, otherwise proceed to step 7.

   If you chose this option, then select from the provided list and continue to step 10.

7. (Optional) Click **Languages...**, if you wish to change the default language used in generating the log entries, otherwise proceed to step 8.

   If you chose this option, then place a check mark in the box next to the language, or languages, you wish to add. Continue to step 8.

8. (Optional) Click **Schedules...**, if you wish to change the default time at which this request is to be run. Perform either a or b, following.

   a. Click **Apply a Saved Schedule...** to choose from a list of predefined schedules.

   b. Chose an option from the Run the Job... list.You can set how often, and when, you wish the start request to run. Depending on which radio button you check, a number of additional fields display so that you may specify exact dates and times.

9. (Optional) Click **Options...**, to specify to whom, and in what language, you wish a notification to be sent. You can also direct the output log files to be sent to a specific printer.

10. Click **Submit** to initiate the start process and to close the dialog box.

> **Note:** Each request generates a Request ID. You can use this Request ID to view the request status and the log entries through the View Requests form window, available through **Concurrent > View Requests**.

### Guidelines

**Options:** The table following lists the available choices. This option indicates which adapters are to be started. You must select from the provided list of values.

| Option | Description |
|---|---|
| ALL | All adapters (for all fulfillment elements) marked AUTOMATIC are started.<br>(You set an adapter to AUTOMATIC through **Administration > Connection Manager**. You can also use the Adapter Properties form window to configure adapter properties.) |
| INCLUDE | Only adapters marked AUTOMATIC that are associated with the designated fulfillment element are started. |
| EXCLUDE | All adapters marked AUTOMATIC are started, except for those associated with the designated fulfillment element. |
| NONE | No adapters are started. |

**Fulfillment Element:** Select the fulfillment element that is to be included or excluded (see the preceding table) from the application start. You must select from the provided list of values.

**Debug Mode:** Select the debugging mode to be used for the Controller and the dequeuer processes. You must select from the provided list of values.

The table following lists the available options.

| Debug Level | Description |
|---|---|
| 0 | No trace |
| 1 | Minimum trace |
| 2 | Medium trace |
| 3 | Maximum trace |

**Note:** Adapters are started with the debug level configured through the Adapter Properties window.

### Oracle SDP Stop

Application Stop is a concurrent request that is performed by the Oracle Provisioning Concurrent Manager.

You use the Application Stop process to perform the following tasks:

- Stop the entire application, including the Controller, dequeuers and all adapters
- Stop a subset of the adapters

### Prerequisites

Application Start and Stop are mutually exclusive processes. If one is pending, the opposite operation does not proceed until the first finishes. In addition, the Oracle Provisioning Concurrent Manager processes must be up and running.

### Steps

1. In the a Navigator, select **Applications > Stop**.

   The Parameters box opens.

2. Enter a value in the Options field from the drop-down list of values.

   See the Guidelines following for information on these choices.

3. Enter a value in the Fulfillment Element field that is to be included (or excluded) from the application stop process.

   This choice depends on the value that you chose in step 2, preceding. See the Guidelines for details.

4. Enter a value for the Stop Mode.

   See the Guidelines following for information on these choices.

5. Click **OK**.

   The Oracle SDP Stop dialog box becomes available. Parameters that you set in the previous dialog box display in the Parameters field.

6. (Optional) Click **Copy**, if you wish to use stop parameters copied from a prior stop request, otherwise proceed to step 7.

   If you chose this option, then select from the provided list and continue to step 10.

7. (Optional) Click **Languages...**, if you wish to change the default language used in generating log entries, otherwise proceed to step 8.

   If you chose this option, then place a check mark in the box next to the language, or languages, you wish to add. Continue to step 8.

8. (Optional) Click **Schedules...**, if you wish to change the default time at which this request is to be run. Perform either a or b, following.

   a. Click **Apply a Saved Schedule...** to choose from a list of predefined schedules.

   b. Chose an option from the Run the Job... list.You can set how often, and when, you wish the start request to run. Depending on which radio button you check, a number of additional fields display so that you may specify exact dates and times.

9. (Optional) Click **Options...**, to specify to whom, and in what language, you wish a notification to be sent. You can also direct the output log files to be sent to a specific printer.

10. Click **Submit** to initiate the stop process and to close the dialog box.

> **Note:** Each request generates a Request ID. You can use this Request ID to view the request status and the log entries through the View Requests form window, available through **Concurrent > View Requests**.

### Guidelines

**Options:** The table following lists the available choices. This option indicates which adapters are to be stopped. You must select from the provided list of values.

| Option | Description |
|---|---|
| ALL | All adapters, for all fulfillment elements, dequeuers and the Controller are stopped. This option signifies the complete application stop. |
| INCLUDE | Only adapters marked AUTOMATIC that are associated with the designated fulfillment element are stopped. |
| EXCLUDE | All adapters marked AUTOMATIC are stopped, except for those associated with the designated fulfillment element. |

**Fulfillment Element:** Select the fulfillment element that is to be included or excluded (see the preceding table) from the application stop. You must select from the provided list of values.

**Stop Mode:** Select the mode to be used. You must select from the provided list of values.

The table following lists the available options.

| Debug Level | Description |
|---|---|
| NORMAL | This specifies the standard shutdown mode. Depending on the choice that you made under Options, this could be the entire system, or only a subset of it.<br>For example, if you select the NORMAL mode, and the Options value is set to ALL, then the following occurs:<br>▫ If there are no adapters still processing orders, then all the dequeuers and the Controller are stopped.<br>▫ If some of the adapters are still processing orders, all the dequeuers stop, except for the Event Manager. Once an adapter has finished processing an order, the Event Manager shuts the adapter down and a database job shuts down the Event Manager and Controller. |
| ABORT | **Specifies an abnormal mode of shutdown. All the running processes (as specified in the Options selection) are terminated at the operating system level.** |

## Provisioning Procedure Macros

Oracle Provisioning comes with a set of predefined constructs that can be used to build procedures. These constructs are.

- SEND
- SEND_HTTP
- LOGIN
- RESPONSE_CONTAINS
- GET_RESPONSE
- GET_PARAM_VALUE
- NOTIFY_ERROR

### SEND

The SEND macro accepts a command string as an argument. This command string is sent to the fulfillment element as the command to be executed. You use the SEND macro to "send" commands to the network elements.

### Signature

The signature of the SEND construct is as follows:

```
SEND(<command > <varchar2>, <encrypt flag> <varchar2>, <prompt (optional)>
<varchar2>, <err code (out)> <number>, <error string (out)> <varchar2>)
```

The application stores all the transactions sent and received, to and from the fulfillment elements in its audit trail. However, it also provides a mechanism that allows you not to store, or write sensitive information contained in the service order request to the audit trail, if desired. Responses from fulfillment elements may also carry the same sensitive information.

Therefore:

- If you set the encryption flag to Y, then you are disabling the logging of all responses from the fulfillment elements.
- If you set the encryption flag to N, then you are enabling the normal logging of responses from the fulfillment elements.

### Syntax

The usage of the construct is defined as follows:

```
SEND(
<command> string which contains the command to be sent,this is where you specify
your service parameters in $variable_name format.
<'Y' or 'N'> to indicate whether the command response must be logged into the
audit trail,
<prompt> is an optional field which indicates the prompt at the Network Element
after the execution of <command>>
<a number place holder for the out parameter error_code>,
<a varchar2 place holder for the out parameter error string>
        );
```

### Restrictions

Note the following restrictions on the use of the SEND macro.

1. For parameter values with white spaces, specify the corresponding **$variable_names** in double quotes.
2. Eliminate white spaces between SEND and the "(" following it.

   Correct Usage: SEND(

   Incorrect Usage: SEND (
3. Do not use SEND inside comments.

### SEND_HTTP

The SEND_HTTP macro accepts a command string as an argument. This command is then sent to the fulfillment element as the command to be executed using the HTTP protocol. Typically, in this scenario, the command is an URL.

### Signature

The signature of the construct is as follows:

```
SEND(
 <a string which contains the URL to be sent>  ,
<'Y' or 'N' indicator which indicates whether the URL should be logged into the
audit trail>,
<a number place holder for the out parameter error_code>,
<a varchar2 place holder for the out parameter error string>
);
```

### Restrictions

Note the following restrictions on the use of the SEND_HTTP macro.

- Eliminate white spaces between SEND_HTTP and the "(" following it.
- Incorrect Usage: SEND_HTTP (
- Correct Usage: SEND_HTTP(

### LOGIN

The LOGIN macro accepts a command string as an argument. It uses the argument to initiate a connection. Use this macro to initiate a connection to a fulfillment element which has an interactive interface (for example, Telnet).

> **Note:** The LOGIN construct should appear before the first SEND construct in the procedure.

### Signature

The signature of the LOGIN construct is as follows:

```
LOGIN(
<a string which contains the command to be sent>  ,
< a string which contains the command response expected>,
<a number place holder for the out parameter error_code>,
<a varchar2 place holder for the out parameter error string>
);
```

### SEND (Connection Procedure)

This SEND macro accepts a command string as an argument. It uses the argument to send the command set (connect) to the fulfillment element. One of the connection commands is executed.

### Signature

The signature of the LOGIN construct is as follows:

```
SEND(
 <a string which contains the command to be sent>  ,
< a string which contains the command response expected>,
<a number place holder for the out parameter error_code>,
<a varchar2 place holder for the out parameter error string>
);
```

### Restrictions

Note the following restrictions on the use of the LOGIN macro.

1. Use the LOGIN macro once only before any SEND construct.
2. Use the LOGIN macro only in a connect procedure.
3. Do not use order, line, work item, and fulfillment action as service parameters. The $ variables are fulfillment element attributes, and, as such, can not be used as service parameters.

## RESPONSE_CONTAINS

The RESPONSE_CONTAINS procedure macro is a PL/SQL function that compares a user-defined string against the string returned by a fulfillment element in response to the last command that it received. The macro returns TRUE if the two strings match, otherwise it returns FALSE.

### Signature

The signature of the construct is as follows:

```
RESPONSE_CONTAINS(
    <a string which contains the string to be matched>
  )
    return BOOLEAN;
```

### Restrictions

Note the following restrictions on the use of the RESPONSE_CONTAINS macro.

1. Do not use $ variables.
2. Eliminate white spaces between RESPONSE_CONTAINS and the "(" following it.
   - Incorrect Usage: RESPONSE_CONTAINS (
   - Correct Usage: RESPONSE_CONTAINS(

### GET_RESPONSE

The GET_RESPONSE macro returns the response string from the fulfillment elements for the **last** command sent.

#### Signature

The signature of the construct is as follows:

```
GET_RESPONSE return VARCHAR2
```

#### Syntax

The usage of the construct is defined as follows:

```
  err_code number;
  err_str     varchar2(400);
  lv_user varchar2(80);
BEGIN
  lv_user := GET_PARAM_VALUE('$WI.user');
SEND('AddProfile -h $WI.hostname -p $FA.dbport -u $WI.user', 'Y', err_code, err_
str);
IF RESPONSE_CONTAINS('profile already exist') THEN
  NOTIFY_ERROR('Error when provisioning Cisco Secure HA Server. Subscriber ' ||
lv_user || ' already exists','A');
  RETURN;
END IF;
EXCEPTION
  WHEN OTHERS THEN
  RAISE;
END;
```

#### Restrictions

Note the following restrictions on the use of the GET_RESPONSE macro.

1. Do not use $ variables.
2. Eliminate white spaces between RESPONSE_CONTAINS and the "(" following it.
   - Incorrect Usage: GET_RESPONSE (
   - Correct Usage: GET_RESPONSE(

### GET_PARAM_VALUE

Use the GET_PARAM_VALUE procedure macro to obtain the value of the parameter during run-time.

#### Signature

The signature of the construct is as follows:

```
GET_PARAM_VALUE (<parameter name> <varchar2>) return VARCHAR2
```

- The <parameter name> is the name of the parameter in *$* format.
- The value returned by the function *can* be *NULL*.

When using this macro, specify the parameters as a *$* variable. For example, parameter *TN (*if used inside a provisioning procedure), is specified as *$ORDER.TN*.

#### Restrictions

Note the following restriction on the use of the GET_PARAM_VALUE macro.

- Eliminate white spaces between GET_PARAM_VALUE and the "(" following it.
- Incorrect Usage: GET_PARAM_VALUE (
- Correct Usage: GET_PARAM_VALUE(

### NOTIFY_ERROR

The NOTIFY_ERROR procedure macro indicates an abnormal exit point from the procedure. The provisioning of a service order request at a fulfillment element has an ERROR status.

This macro appends or replaces a response initiated by the fulfillment element / network element. This is the input string.

### Signature

The signature of the construct is as follows:

```
NOTIFY_ERROR(<user's string> <varchar2>, <mode> <char>)
```

- The <user's string> contains a user-defined friendly message
- The <mode> flag is either **A** or **R**.

  **A** indicates that the user message is to be appended to the response string coming from the fulfillment element.

  **R** indicates that the user message is to be replaced by the response string coming from the fulfillment element.

### Guidelines

1. You use this macro under the following conditions:
   - You want to identify errors coming from the fulfillment element during the provisioning process.
   - You want to identify errors when exiting from the provisioning procedure.
2. You use the NOTIFY_ERROR macro to abort a provisioning procedure.
3. You must follow every NOTIFY_ERROR instance in the procedure by a RETURN statement. This is the only way to abort the execution of the provisioning procedure.
4. The Service Delivery Platform uses the <user's string> to display the provisioning errors of a failed Service Order Request. Make this name user-friendly. This name provides the ability to translate cryptic network element error messages into those that are more user friendly.

## Example Service Delivery Platform Procedures

Following are several example workflow procedures:

- Provisioning procedure
- Connect procedure

### Provisioning Procedure

The following is an example of a Provisioning Procedure

```
DECLARE
err_code number;
err_str     varchar2(400);
lv_user varchar2(80);
BEGIN
lv_user := GET_PARAM_VALUE ('$user');
  SEND('AddProfile -h $hostname -p $dbport -u $user', 'Y', err_code, err_str);
  IF RESPONSE_CONTAINS ('profile already exist') THEN
     NOTIFY_ERROR('Error when provisioning Cisco Secure HA Server. Subscriber '
|| lv_user || ' already exists','A');
RETURN;
 END IF;
EXCEPTION
WHEN OTHERS THEN
   RAISE;
END;
```

### Connect Procedure

The following is an example of a Connect Procedure

```
DECLARE
err_code number;
err_str     varchar2(400);
lv_user varchar2(80);
BEGIN
LOGIN('telnet $ip_address', 'login:', err_code, err_str);
SEND('$username', '$Password:', err_code, err_str);
SEND('$Password', '$', err_code, err_str);

EXCEPTION
WHEN OTHERS THEN
   RAISE;
END;
```

# Workflows

## Processes

### Service Activation and Provisioning Functions

Service Delivery encompasses service provisioning and activation / deactivation (temporary and permanent). This is one of the functional domains (fulfillment) provided by Oracle's Service Delivery Platform (SDP).

Order fulfillment involves accepting a request and fulfilling it in one of two ways:

- Performing the requested functions internally within SDP
- Directing external systems to perform identified functions

Handling the results of the fulfillment process is part of the functionality of the fulfillment software that SDP provides. The fulfillment software capability includes:

- Service activation and provisioning management
- SDP supports management for fallout from the fulfillment process,
- Management of the fulfillment process
- Detection and resolution of jeopardy conditions (accomplished through user interfaces and Management Centers).

### Order Fulfillment Process

The SDP performs the following business functions of order fulfillment:

- Order Capture
- Order Manipulation, Processing
- Order Visibility / Status
- Order Fulfillment / Provisioning / Repair
- Service Order Assurance (projected for end of Year 20000

The SDP interfaces to any Billing System available at a telecommunications service provider's site.

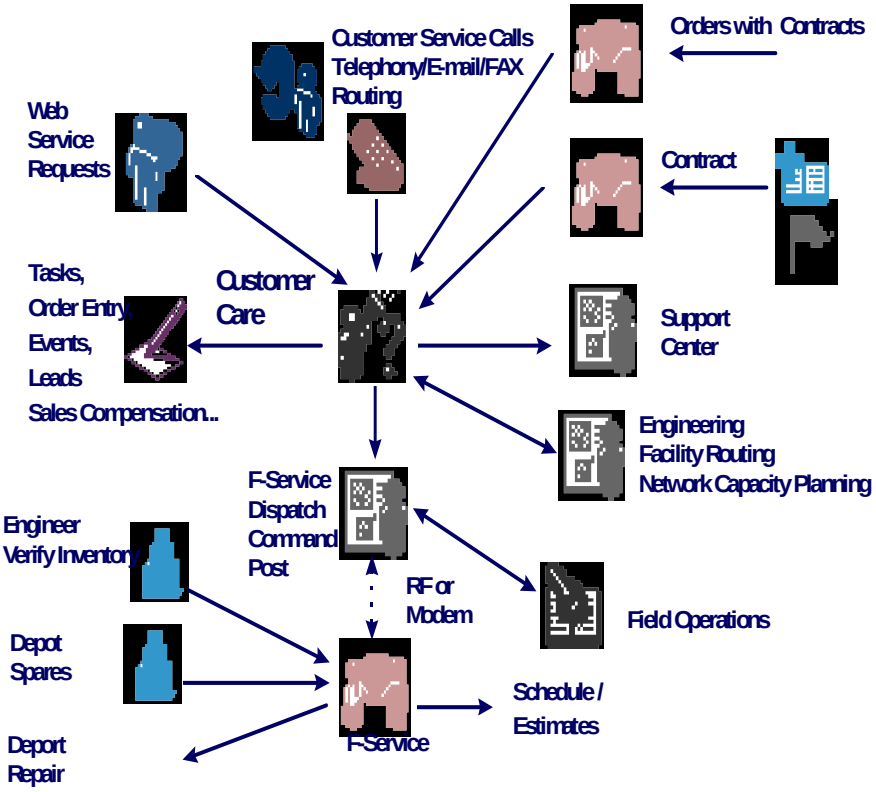
### Order Provisioning and Activation

SDP will segment service orders received from different Customer Care Order / Entry System applications into the required set of work tasks to fulfil an order.

For example, early in the process SDP can request:

- The assignment of physical inventory.
- The configuration and activation of services on the switches.
- When all work tasks have been completed, SDP will notify the Billing system which in turn will activate the commencement of billing.
- Customers Service Representatives will then be able to query the status of the service request orders.

## Service Order Request Process Flow



## Service Order Request Process

### Sales Functions and Processes

The following are sales functions and processes

- Learn about a customer's needs.
- Educate a customer on available services and pricing.
- Match the customer's requirements to the product offerings.
- Arrange for the appropriate options.

### Customer Interface Management

The process supported by the Customer Care, Order Entry systems are:

- Receive and record a contact person with a customer profil.e
- Make inquiries to the other sub-systems (inventory, billing, etc.).
- Monitor and control status of inquiries on a service order request and escalate a problem.
- Ensure a consistent image an representation of service order request and secure the information.

### Order Handling / Order Manager

The process supported by the Order Entry, Order Manager are:

- Accept orders
- Determine pre-order feasibility
- Prepare price estimate
- Develop order plan
- Perform credit check
- Request customer deposit
- Initiate service installation
- Establish SLA terms
- Track order status
- Complete order, notify customer
- Initiate billing process

## Handling Failed Service Order Requests

### Problem Handling

The Service Delivery Platform has a Fallout Management Center (FMC) module. The Fallout Management Center is part of the SDP's Demand Repair Center, Jeopardy Management Center. The FMC functions are:

- Receive trouble notifications.
- Determine cause and resolve / refer the notification message.
- Track progress of a trouble resolution.
- Initiate action to reconfigure a service, if needed.
- Generate a notification, forward the notification to the Trouble Ticket system and / or Work Force Management.s.
- Confirm that the trouble has been cleared.
- Notify the Customer Care system when the trouble is cleared.
- Report on the completion of the trouble.

### Ordering Problem Resolution of a Service at the Order Level

Problems are resolved in the following ways:

- Isolate and resolve all service problems.
- Identify chronic failures & provide performance data.

## Workflows

### Using Timers in Workflow

You can build timer functionality into your user-defined Workflow by using a set of drag-and-drop activities provided by Oracle Provisioning. These activities are part of the SDP Standard item type.

The table following lists the activities that can be associated with timers in Workflow:

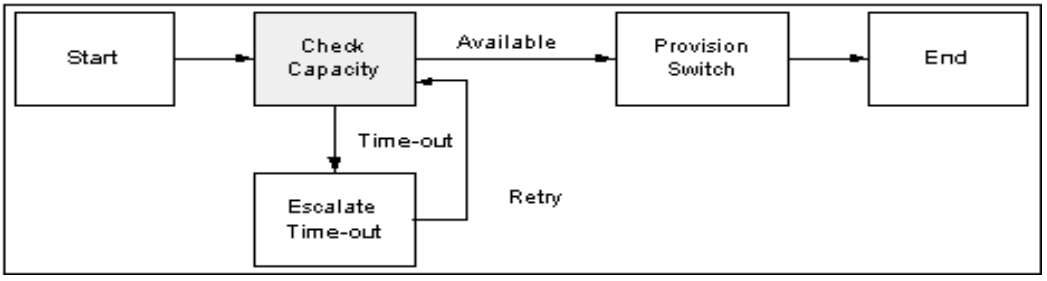
| Function Name | Description |
|---|---|
| Deregister | Remove all timers for a given order |
| Fire Timer | Function to start either a process or activity timer |
| Send Message | Function that sends a message which has a timer associated with it |
| Start Related Timers | Function to start either an activity timer or a timer associated with a message |
| Subscribe to Acknowledgements | Function to subscriber for possible results from the Send Message function |
| Get Jeopardy Flag | Checks whether or not a jeopardy timer exists for a given order |
| Get Timer Status | Returns the status of a timer |
| Recalculate all timers | Recalculates the delay and interval elements for all timers associated with the given the Work Item name, Instance ID, Order ID, or the Fulfillment Action Instance ID. |
| Remove Timer | Deletes the timer with the given name |

### Activity Timers in Oracle Provisioning

An Activity Timer is used in conjunction with a workflow activity. The purpose of an Activity Timer is to provide the ability to set a pre-defined period of time within which a workflow activity must be completed.

For example, a workflow activity is defined to check the network capacity for a requested data service. The time period to perform the workflow activity is set to 30 minutes. An Activity Timer is then associated with the workflow activity to handle the required business logic if the 30 minute time frame for the activity is exceeded.

***Example Workflow Activity Timer***

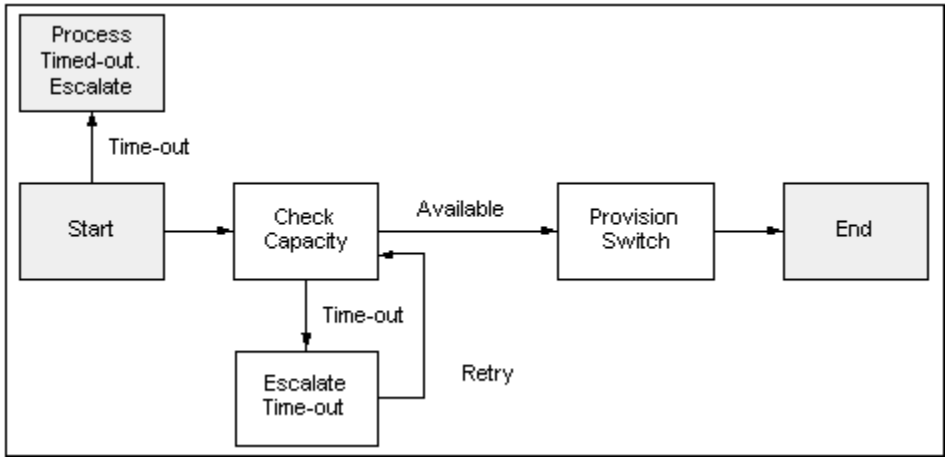


### Process Timers in Oracle Provisioning

A Process Timer is used in conjunction with a Workflow process. The purpose of a Process Timer is to provide the ability to set a pre-defined period of time within which a Workflow process must be completed.

For example, a workflow process is defined to provide a requested data service. The time period to perform the Workflow process is set to one business day. A Process Timer is then associated with the workflow process to handle the required business logic if the one day time frame for the process is exceeded.

***Example Workflow Process Timer***



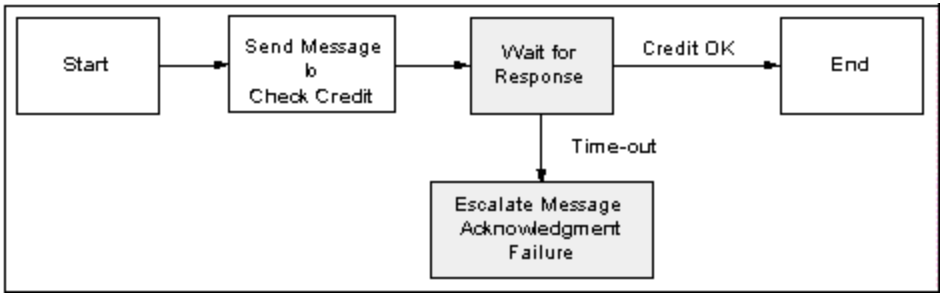
### Message Timers in Oracle Provisioning

A Message Timer is used in conjunction with a message. The purpose of a Message Timer is to provide the ability to set a pre-defined period of time within which a message acknowledgment must be received.

For example, a workflow process is defined to set up an account for a network service. A message is sent to the associated network service administrator. If a reply is not received back with in 30 minutes, then the timer message is published and picked up by the workflow activity that is **Waiting for the Acknowledgments**. The acknowledgment may be a message reply or a timer in which the workflow follows a path based on the message received.

The Waiting For Acknowledgments activity subscribes to events. An event can be configured to include an acknowledgment for the message sent out, as well as a timer. Depending on the message received, the workflow progresses.

***Example Message Timer***



### Workflow in Oracle Provisioning

Workflows are processes that execute during the main provisioning process to process and fulfill the order. Workflows can be run before starting the service provisioning process.

You use the Oracle Workflow Builder to configure the workflows to be executed during the provisioning process. The workflow can be invoked for a work item or a fulfillment action.

The main provisioning workflow waits until all sub-processes complete their activities before continuing to fulfill the order.

If an error occurs during the main provisioning process, you can intervene manually and launch appropriate workflows as the situation dictates. You can launch any workflow process in Oracle Provisioning, as necessary.

### The Workflow Builder

Refer to the *Oracle Workflow Guide* for details of the workflow builder.

### Workflow Processing of a Service Order Request

During the execution of a service order, the line items associated with it are processed by workflow. Which fulfillment procedures execute depend on the type and software generic of the fulfillment element configured in Oracle Provisioning.

- The mapping between fulfillment procedure and fulfillment element is specified in the Service Creation Manager.
- The fulfillment procedure involves sending commands to a fulfillment element, receiving, and analyzing responses to determine success or failure of commands.
- The procedures are written in PL/SQL which has been extended with open APIs.
- A watchdog workflow detects the termination of adapter processes and manages their recovery.
- When all adapters for a given fulfillment element are shutdown, suspended, disconnected or goes down, all service order requests for that fulfillment element are queued until the adapter is up again and ready to process the next fulfillment action in the queue.

### Steps Involved in Processing a Service Order Request

- A service order consists of many line items. A line item can be either a service or a package.
- Each line item can be mapped to multiple work items.
- Each work item is mapped to a set of fulfillment actions via different means. (For example, this could be either a static or dynamic fulfillment action, workflow, or a workflow procedure.)
- Each fulfillment action invokes one fulfillment procedure at runtime depending on the type of fulfillment element.
- Each procedure is executed on one fulfillment element.
- The identity of the fulfillment element to be provisioned is determined through routing rules implemented through user-defined PL/SQL procedure.
- Error conditions detected during provisioning cause the workflow to send error notifications.
- The workflow waits for the notification response.
- The response indicates how the error condition is to be handled.

### The Workflow Monitor

The Workflow Monitor utility provides a graphical interface to track a service order's path in Oracle Provisioning. When launched, it opens in a web browser screen. The browser screen displays the runtime path for the order within the Oracle Service Delivery Platform.

Using the graphical interface, you can:

- View each activity performed on an order.
- View the route of an order from the beginning to the end.
- Retrieve statistical information and branching activities for each node displayed on the workflow path.

### Workflow Information Types

Oracle Provisioning workflow provides the following categories of information:

- Definition: The definition of the actual activity as defined in the Workflow. This also includes the description of the activity, its type and its results.
- Usage: The location, start/end status, performer, and time-out status information.
- Status: The status of execution of a particular activity, the begin and end dates for the activity.
- Notification: The recipient, the current status, the date sent, due date and end date for a given notification.

### Procedure Builder

Oracle Provisioning provides a procedure builder tool that you use to write PL/SQL based procedures. PL/SQL procedures must be defined before processing a service order request.

The procedure builder comes with a library of constructs (building blocks) that you can use to write procedures. These procedures, then, are used during the various phases of provisioning a service order

---

**Note:** It is expected that the user-defined procedures are written by system experts who have knowledge of the provisioning syntax and service parameters required for the fulfillment element. They must also have good knowledge of PL/SQL.

---

### Procedure types

The different procedures types are:

- Connect and disconnect: The connect procedure is used to connect Oracle Provisioning to a fulfillment element when the Oracle Provisioning application is started. Similarly the disconnect procedure disconnects Oracle Product Name from a fulfillment element.

  You can choose from a set of PL/SQL constructs which can be used to write connection/disconnection procedures for multiple fulfillment elements.

- Fulfillment element routing: The Locate Fulfillment Element procedure returns the fulfillment element that should be provisioned.

- Dynamic work item-fulfillment action mapping: If you choose the work item type to be of type dynamic, you must use the procedure builder to define the procedure that is executed when the order is sent to Oracle Provisioning. This procedure specifies the fulfillment actions to be executed for the work item.

- Work item parameter evaluation: The evaluation procedure evaluates the parameter values for the work item parameters.

- Fulfillment action parameter evaluation: The evaluation procedure evaluates the parameter values for the fulfillment action parameters.

- User-defined workflow: This procedure is used to launch a user defined workflow. A workflow can consist of multiple procedures to be executed in a pre-set sequence.

- Dynamic service to work item mapping: This procedure specifies at runtime which work items will be executed to fulfill the service.

- Evaluate all fulfillment action parameters: This evaluation procedure is used to evaluate all the fulfillment action parameters together.

- Fulfillment procedure: This procedure specifies the provisioning commands that need to be executed in fulfilling a service order.

# Testing an Implementation Project

## Submitting a Test Order

You use the Quick Order Entry window to create a new service order for testing purposes.

### Prerequisites

None

### Steps

1. In the Navigator, choose **Test Center > Test Order**.

   A blank Quick Order Entry form window opens.

2. Select the Order Header tab.

3. Enter an order number.

   - If you want to create a new test order, enter its order number here.
   - If an order is already defined, enter its order number to populate the form window. This information is read-only, and can not be modified.
   - If you do not know the order number for an existing order, click **Find Orders** and perform a search for it using the Find Orders dialog box.

4. (Optional) Enter an order version.

5. (Optional) Select Order Type, and choose a value from the drop-down list. (Typically, this is External System Order.)

6. (Optional) Enter customer information.

   If your installation is customized and integrated with a customer care system, then enter a value. Otherwise, ignore the customer fields, they are used for searching purposes only.

7. Enter a value for Provisioning Date.

   This is the date on which porting activation is to take place. Use the Calendar selection tool, if desired. You can also enter values for Customer Required Date, and Due Date, if you want these fields to be used for search queries.

8. (Optional) Click **Order Params** to open the Order Parameters selection dialog box.

   If desired, select an order parameter from the drop-down list of values, and enter a value for it in the Parameter Value field. Click **OK** to save and close the dialog box.

9. (Optional) Select the Line Items tab.

   You use this dialog box to set information about services associated with this order.

   a. Select the Details tab.

      Depending on the order type that you selected in step 5, different fields display.

      If you selected External System Order, then fields pertaining to services display.

      If you selected Number Portability System Order, then fields pertaining to work items display.

   b. Complete the fields in the Details tab as necessary.

   c. Complete the fields in the Sequence tab as necessary.

   d. Click **Line Parameters** to open the Line Item Parameters dialog box.

      Use this dialog box to associate a line item parameter and value with the order. Click OK to save your entries and close the dialog box.

10. Click **Submit** to generate a test order.

# Considerations for Future Upgrade Paths

The following are considerations for future System-Level upgrades:

- System Profiles
- Enabled Workflows
- Employees and Security
- Multicurrency capabilities
- Customization Issues

# A

# Public APIs

## PROCEDURE Process_Order

### Specification

**API Name:**
Process_Order

**Type:**
Public

**Purpose:**
This API is used for submitting a service order to SDP.

**Pre-Requisites:**
None.

**Parameters:**

```
       IN              :       p_api_version:  NUMBER  Required
                               p_init_msg_list:        VARCHAR2        Optional
                                       Default = FND_API.G_FALSE
                               p_commit:               VARCHAR2        Optional
                                       Default = FND_API.G_FALSE
                                       When calling this API via a database
link, this
                                       parameter must be FND_API.G_FALSE
                               p_validation_level      NUMBER  Optional
                                       Default = FND_API.G_VALID_LEVEL_FULL
                               p_order_header: XDP_TYPES.ORDER_HEADER Required
                                       Order header information which requires
the
                                       following atrribute values to be
supplied:
                                       order_number:
                                               The order identifier which is
assigned by the
                                               calling system
                                       order_version:
                                               The version of the order.  This
attribute
                                               can be NULL
                                       provisioning_date:
```

```
                                                The date this order is supposed
to begin
                                                provisioning.  If the value is
not supplied then
                                                it will be default to the
sysdate.
                        (Continued on next page)

(Continued from previous page)

                                             order_action:
                                                The provisioning action which
can be applied
                                                to all the lines in the order.
This attribute is
                                               optional.
                                            jeopardy_enabled_flag:
                                                The flag indicates whether the
jeopardy analysis
                                                should be enabled for the order
or not.  The user
                                                can then use this flag combine
with other order
                                                information in the post process
user hook package
                                                to determine whether to start a
jeopardy timer or
                                                not.  This attribute is
optional.
                                    p_order_parameter: XDP_TYPES.ORDER_PARAMETER_
LIST

  (Continued on next page)

(Continued from previous page)
Required
                                            The parameters that can be accessed by
all
                                            the order lines. The list can be empty.
                                   p_order_line_list: XDP_TYPES.ORDER_LINE_LIST
Required
                                            The list of order line items which
requires
                                            the following attribute values to be
supplied:
```

line_number:
The index number of the current line
line_item_name:
The name of the service item
version:
The version of the service item
action:
The provisioning action of the current
line. If this
value is not supplied, SDP will default
it to the order
action
provisioning_date:
The date this order line is scheduled to
be provisioning. If the value is not
supplied SDP will
default it to order provisioning date.
provisioning_sequence:
SDP uses this attribute to determine
dependency between
order lines. If the value is not
supplied SDP will
assume there is not dependency for this
line.

(Continued on next page)

(Continued from previous page)

p_line_parameter_list: XDP_TYPES.LINE_PARAM_
LIST Required
The list of the parameters for each
order line. The list
can be empty. For every record in the
list, the
following attribute values to be
supplied:
line_number:
The line number of this parameter is
associated with
parameter_name:
The name of the parameter
parameter_value:
The value of the parameter. It can be
null.
parameter_ref_value:
The reference value of the parameter.

```
This
                                        attribute is optional.

        OUT             :       x_return_status:        VARCHAR2(1) Required
                                        The caller must examine this parameter
value
                                        after the call is completed.  If the
value is
                                        FND_API.G_RET_STS_SUCCESS, the caller
routine
                                        must do commit, otherwise, the caller
routine
                                        must do rollback.
                                x_msg_count:            NUMBER
                                x_msg_data:                     VARCHAR2(2000)
                                x_sdp_Order_id:         NUMBER
                                The internal order ID which is assigned by SDP
                                when an order is successfully submitted to SDP
```

**Version:**

Current version 11.5

**Notes:**

This API is used for upstream ordering system to submit a service order to SDP. If the customer wishes to perform order dependency and jeopardy analysis, he or she can put the business logic in the post process API under the customer hook package which will be supported by SDP per CRM coding standard.

**PROCEDURE Process_Order(**

```
        p_api_version   IN      NUMBER,
        p_init_msg_list IN      VARCHAR2 := FND_API.G_FALSE,
        p_commit                IN      VARCHAR2 := FND_API.G_FALSE,
        p_validation_level      IN      NUMBER :=
                                                FND_API.G_VALID_LEVEL_
FULL,
        x_RETURN_STATUS         OUT VARCHAR2,
        x_msg_count                     OUT NUMBER,
        x_msg_data                      OUT VARCHAR2,
        P_ORDER_HEADER          IN  XDP_TYPES.ORDER_HEADER,
        P_ORDER_PARAMETER       IN  XDP_TYPES.ORDER_PARAMETER_LIST,
        P_ORDER_LINE_LIST       IN  XDP_TYPES.ORDER_LINE_LIST,
        P_LINE_PARAMETER_LIST   IN  XDP_TYPES.LINE_PARAM_LIST,
        x_SDP_ORDER_ID             OUT NUMBER);
```

## PROCEDURE Cancel_Order

### Specification

**API Name:**
Cancel_Order

**Type:**
Public

**Purpose:**
This API is used for canceling a service order

**Pre-Requisites:**
None.

**Parameters:**

```
        IN              :       p_api_version:  NUMBER  Required
                                p_init_msg_list:        VARCHAR2        Optional
                                        Default = FND_API.G_FALSE
                                p_commit:               VARCHAR2        Optional
                                        Default = FND_API.G_FALSE
                                        This API is an autonomous routine which
                                        handles the database transaction
independently.
                                        The value of p_commit parameter will be
ignored.
                                p_validation_level      NUMBER  Optional
                                        Default = FND_API.G_VALID_LEVEL_FULL
                                p_sdp_Order_id:         NUMBER Required
                                       The internal order ID which was assigned
by SDP
                                       when an order was successfully submitted
to SDP
                                p_caller_name:          VARCHAR2  Required
                                       The name of the user who is calling this
API

        OUT             :       x_return_status:        VARCHAR2(1) Required
                                       The execution status of the API call.
                                x_msg_count:            NUMBER
                                x_msg_data:                     VARCHAR2(2000)
```

**Version:**

Current version 11.5

**Notes:**

This API is used for upstream ordering system to cancel a service order which was submitted to SDP previously.

**PROCEDURE Cancel_Order(**

```
        p_api_version   IN      NUMBER,
        p_init_msg_list IN      VARCHAR2 := FND_API.G_FALSE,
        p_commit                IN      VARCHAR2 := FND_API.G_FALSE,
        p_validation_level      IN      NUMBER :=
                                                FND_API.G_VALID_LEVEL_
FULL,
        x_RETURN_STATUS         OUT VARCHAR2,
        x_msg_count                     OUT NUMBER,
        x_msg_data                      OUT VARCHAR2,
        P_SDP_ORDER_ID          IN NUMBER,
        p_caller_name           IN VARCHAR2 );
```

## PROCEDURE Process_DRC_Order

### Specification

```
REM      Process_DRC_Order
REM      Public
REM      API for processing a DRC order in a synchronous mode
REM      None.
REM
REM     IN              :       p_api_version                   IN NUMBER
Required
REM                             p_init_msg_list         IN VARCHAR2     Optional
REM                                     Default = FND_API.G_FALSE
REM                             p_commit                        IN VARCHAR2
Optional
REM                                     Default = FND_API.G_FALSE
REM                             p_validation_level              IN NUMBER
Optional
REM                                     Default = FND_API.G_VALID_LEVEL_FULL
REM                             P_WORKITEM_ID           IN  NUMBER   Required
                                        The internal ID of the work item to be

executed
REM                             P_TASK_PARAMETER        IN XDP_TYPES.ORDER_
                                         PARAMETER_LIST
                                        The list of parameters for the request
```

**API Name:**

Cancel_Order

**Type:**

Public

**Purpose:**

This API is used for canceling a service order

**Pre-Requisites:**

None.

**Parameters:**

```
        OUT             :       x_return_status         OUT     VARCHAR2(1)
                                        The execution status of the API call.
                                x_msg_count                     OUT     NUMBER
                                x_msg_data                      OUT
VARCHAR2(2000)
                                x_sdp_Order_id          OUT NUMBER
                                        The internal order ID which is assigned
by SDP
                                        when the request is fulfilled
```

**Version:**

Current version 11.5

**Notes:**

This API is used for the test center to execute a work item synchronously. The process flow is as followed:

1. Check if the work item can be executed synchronously. The condition is that the FA mapping type of the work item must be either STATIC or DYNAMIC. This API does not invoke any workflow. It does not use any OP process queue either.
2. Create a dummy service order in SDP for tracking purpose only. The internal order ID will be returned to the caller after the call is completed.
3. Find out all the FAs which have been mapped to this work item per configuration.
4. For each FA, find out which FE it will be executed upon.
5. Find the available adapter for the given FE. The usage code for the adapter must be TEST.
6. Execute the appropriate Fulfillment Procedure.
7. Return when all the FAs have been executed.

**PROCEDURE Process_DRC_Order(**

```
        p_api_version   IN      NUMBER,
        p_init_msg_list IN      VARCHAR2 := FND_API.G_FALSE,
        p_commit                IN      VARCHAR2 := FND_API.G_FALSE,
        p_validation_level      IN      NUMBER :=
                                                FND_API.G_VALID_LEVEL_
FULL,
        x_RETURN_STATUS         OUT VARCHAR2,
        x_msg_count                     OUT NUMBER,
        x_msg_data                      OUT VARCHAR2,
        P_WORKITEM_ID           IN  NUMBER,
        P_TASK_PARAMETER        IN XDP_TYPES.ORDER_PARAMETER_LIST,
        x_SDP_ORDER_ID          OUT NUMBER);
```

# B

# SDP Parameters

## SDP Parameters

These parameters are provided with the applications and can be utilized at order, line item, work item, and fulfillment action levels. New parameters can be added to your configuration. Proceeded parameters should never be modified or deleted.r

| Internal Name | Display Name | Description |
|---|---|---|
| 'MESSAGE_VERSION' | 'MessageVersion' | 'Message Version' |
| 'REFERENCE_ID' | 'ReferenceId' | 'ReferenceIdentification' |
| 'OPP_REFERENCE_ID' | 'OppReferenceId' | 'OppReferenceIdentification' |
| 'SUBSCRIPTION_TN' | 'SubscriptionTN' | 'SubscriptionTelephoneNumber' |
| 'STARTING_NUMBER' | 'StartingNumber' | 'Starting Number' |
| 'ENDING_NUMBER' | 'EndingNumber' | 'Ending Number' |
| 'PORTING_ID' | 'PortingId' | 'PortingIdentification' |
| 'NEW_SP_DUE_DATE' | 'NewSPDueDate' | 'NewServiceProviderDueDate' |
| 'DONOR_SP_ID' | 'DonorSPId' | 'Donor Service Provider Identification' |
| 'RECIPIENT_SP_ID' | 'RecipientSPId' | 'Recipient Service Provider Identification' |
| 'ROUTING_NUMBER' | 'RoutingNumber' | 'Routing Number' |
| 'MESSAGE_ID' | 'MessageId' | 'MessageIdentification' |
| 'CUSTOMER_ID' | 'CustomerId' | 'CustomerIdentification' |
| 'CUSTOMER_NAME' | 'CustomerName' | 'Customer Name' |
| 'ADDRESS_LINE1' | 'AddressLine1' | 'Address Line1' |
| 'ADDRESS_LINE2' | 'AddressLine2' | 'Address Line2' |
| 'ZIP_CODE' | 'Zip Code' | 'Zip Code' |
| 'CITY' | 'City' | 'City' |
| 'PHONE' | 'Phone' | 'Phone' |
| 'FAX' | 'Tax' | 'Tax' |
| 'EMAIL' | 'Email' | 'Email' |
| 'CONTACT_NAME' | 'ContactName' | 'Contact Name' |
| 'CONTACT_DEPT' | 'ContactDept.' | 'Contact Department' |

| | | |
|---|---|---|
| 'CUSTOMER_CONTACT_REQ_FLAG' | 'Customer Contact Req Flag' | 'CustomerContactRequiredFlag' |
| 'OLD_SP_DUE_DATE' | 'OldSPDueDate' | 'OldServiceProviderDueDate' |
| 'ORDER_RESULT' | 'OrderResult' | 'Order Result' |
| 'ORDER_REJECT_CODE' | 'OrderRejectCode' | 'Order Reject Code' |
| 'ORDER_REJECT_EXPLN' | 'OrderRejectExplain' | 'OrderRejectExplanation' |
| 'OLD_SP_CUTOFF_DUE_DATE' | 'Old SP Cutoff Due Date' | 'Old Service Provider Cutoff Due Date' |
| 'RETAIN_TN_FLAG' | 'Retain Subscription TN Flag' | 'Retain Subscription Telephone Number Flag' |
| 'RETAIN_DIR_INFO_FLAG' | 'RetainDirInfoFlag' | 'RetainDirectoryInformationFlag' |
| 'PRIORITY' | 'Priority' | 'Priority' |
| 'GEO_AREA_CODE' | 'GeoAreaCode' | 'GeographicAreaCode' |
| 'PRICE_PER_MINUTE' | 'PriceperMinute' | 'Price per Minute' |
| 'PRICE_PER_CALL' | 'PriceperCall' | 'Price per Call' |
| 'PRICE_CODE' | 'PriceCode' | 'Price Code' |
| 'SERVICE_INFO' | 'ServiceInfo' | 'ServiceInformation' |
| 'INVOICE_DUE_DATE' | 'InvoiceDueDate' | 'Invoice Due Date' |
| 'ORDER_PRIORITY' | 'OrderPriority' | 'Order Priority' |
| 'CHARGING_INFO' | 'ChargingInfo' | 'ChargingInformation' |
| 'SENDER_NAME' | 'Sender Name' | 'Sender Name' |
| 'RECIPIENT_NAME' | 'RecipientName' | 'Recipient Name' |
| 'INITIAL_DONOR_SP_ID' | 'InitialDonorSPId' | 'Initial Donor Service Provider Identification' |
| 'CREATION_DATE' | 'CreationDate' | 'Creation Date' |
| 'ORDER_VERSION' | 'OrderVersion' | 'Order Version' |
| 'CLASS_ADDRESS' | 'CLASSAddress' | 'CLASS Address' |
| 'CLASS_SUBSYSTEM' | 'CLASSSubsystem' | 'CLASS Subsystem' |
| 'CNAM_ADDRESS' | 'CNAMAddress' | 'CNAM Address' |
| 'CNAM_SUBSYSTEM' | 'CNAMSubsystem' | 'CNAM Subsystem' |

| | | |
|---|---|---|
| 'ISVM_ADDRESS' | 'ISVMAddress' | 'ISVM Address' |
| 'ISVM_SUBSYSTEM' | 'ISVMSubsystem' | 'ISVM Subsystem' |
| 'LIDB_ADDRESS' | 'LIDBAddress' | 'LIDB Address' |
| 'LIDB_SUBSYSTEM' | 'LIDBSubsystem' | 'LIDB Subsystem' |
| 'WSMSC_ADDRESS' | 'WSMSCAddress' | 'WSMSC Address' |
| 'WSMSC_SUBSYSTEM' | 'WSMSCSubsystem' | 'WSMSC Subsystem' |
| 'RN_ADDRESS' | 'RNAddress' | 'RN Address' |
| 'RN_SUBSYSTEM' | 'RNSubsystem' | 'RN Subsystem' |
| 'EXT_CLASS_ADDRESS' | 'ExtCLASSAddress' | 'External CLASS Address' |
| 'EXT_CLASS_SUBSYSTEM' | 'Ext CLASS Subsystem' | 'Exteral CLASS Subsystem' |
| 'EXT_ISVM_ADDRESS' | 'ExtISVMAddress' | 'External ISVM Address' |
| 'EXT_ISVM_SUBSYSTEM' | 'ExtISVMSubsystem' | 'External ISVM Subsystem' |
| 'EXT_CNAM_ADDRESS' | 'ExtCNAMAddress' | 'External CNAM Address' |
| 'EXT_CNAM_SUBSYSTEM' | 'Ext CNAM Subsystem' | 'External CNAM Subsystem' |
| 'EXT_LIDB_ADDRESS' | 'ExtLIDBAddress' | 'External LIDB Address' |
| 'EXT_LIDB_SUBSYSTEM' | 'ExtLIDBSubsystem' | 'External LIDB Subsystem' |
| 'EXT_WSMSC_ADDRESS' | 'ExtWSMSCAddress' | 'External WSMSC Address' |
| 'EXT_WSMSC_ SUBSYSTEM' | 'Ext WSMSC Subsystem' | 'Exteral WSMSC Subsystem' |
| 'EXT_RN_ADDRESS' | 'ExtRNAddress' | 'External RN Address' |
| 'EXT_RN_SUBSYSTEM' | 'ExtRNSubsystem' | 'External RN Subsystem' |
| 'ERROR_CODE' | 'Error Code' | 'Error Code' |
| 'ERROR_DESCRIPTION' | 'Error Description' | 'Error Description' |
| 'EXT_ROUTING_NUMBER' | 'ExtRouting Number' | 'External Routing Number' |
| 'SP_ID' | 'SP Id' | 'Service Provider Identification' |
| 'AUDIT_NAME' | 'Audit Name' | 'Audit Name' |
| 'SMS_FAILURE_DATE' | 'SMS Failure Date' | 'SMS Failure Date' |
| 'ADAPTER_NAME' | 'Adapter Name' | 'Adapter Name' |

| | | |
|---|---|---|
| 'GEO_AREA_TYPE_CODE' | 'GeoAreaTypeCode' | 'GeographicAreaTypeCode' |
| 'GEO_AREA_NAME' | 'GeoAreaName' | 'GeographicAreaName' |
| 'GEO_AREA_DISPLAY_ NAME' | 'Geo Area Display Name' | 'GeographicAreaDisplayName' |
| 'INTERCONNECT_TYPE' | 'InterconnectType' | 'Interconnect Type' |
| 'FE_NAME' | 'FEName' | 'FulfillmentElementName' |
| 'ORDER_ID' | 'OrderId' | 'OrderIdentification' |
| 'MESSAGE_CODE' | 'MessageCode' | 'Message Code' |
| 'ASSIGNED_SP_ID' | 'AssignedSPId' | 'Assigned Service Provider Identification' |
| 'OWNING_SP_ID' | 'OwningSPId' | 'Owning Service Provider Identification' |
| 'NUMBER_RANGE_CODE' | 'NumberRangeCode' | 'NumberRangeCode' |
| 'EFFECTIVE_DATE' | 'EffectiveDate' | 'Effective Date' |
| 'GEO_INDICATOR' | 'GeoIndicator' | 'GeographicIndicator' |
| 'MOBILE_INDICATOR' | 'MobileIndicator' | 'Mobile Indicator' |
| 'POOLED_FLAG' | 'PooledFlag' | 'Pooled Flag' |
| 'PORTED_INDICATOR' | 'PortedIndicator' | 'Set if the Number Range is Open to Portability' |
| 'SERVICE_TYPE' | 'ServiceType' | 'Service Type' |
| 'STATUS' | 'Status' | 'Status' |
| 'PERMISSIVE_DIAL_ START_DATE' | 'Permissive Dial Start Date' | 'Start Date on which the old and newnumberrangearebothactive.' |
| 'PERMISSIVE_DIAL_END_ DATE' | 'Permissive Dial End Date' | 'End Date after which the old numberrangeisnolongeractive.' |
| 'OLD_NUMBER_RANGE_ CODE' | 'Old Number Range Code' | 'Number range prior to a number range split.' |
| 'NEW_NUMBER_RANGE_ CODE' | 'New Number Range Code' | 'New number range after a number range split.' |
| 'CONVERSION_ PROCEDURE' | 'Conversion Procedure' | 'Procedure to covert an old number to the new number when it has been affected by |
| 'FEATURE_TYPE' | 'FeatureType' | 'Feature Type' |

| | | |
|---|---|---|
| 'FE_ID' | 'FE Id' | 'FulfillmentElementIdentification' |
| 'NUMBER_RANGE_ID' | 'NumberRangeId' | 'NumberRangeIdentification' |
| 'PRIMARY_FLAG' | 'PrimaryFlag' | 'Primary Flag' |
| 'COUNTRY' | 'Country' | 'Country' |
| 'DEPARTMENT' | 'Department' | 'Department' |
| 'INTERNET_ADDRESS' | 'InternetAddress' | 'Internet Address' |
| 'MOBILE' | 'Mobile' | 'Mobile' |
| 'SP_NAME' | 'SP Name' | 'Service Provider Name' |
| 'PAGER' | 'Pager' | 'Pager' |
| 'PAGER_PIN' | 'Pager Pin' | 'Pager Pin' |
| 'SERVED_BY_FLAG' | 'Served by Flag' | 'Served by Flag' |
| 'SP_TYPE' | 'SP Type' | 'Service Provider Type' |
| 'STATE' | 'State' | 'State' |
| 'TIMEZONE' | 'Timezone' | 'Timezone' |
| 'BUSINESS_ROLE' | 'Business Role' | 'Business Role' |
| 'ACTIVATION_START_ DATE' | 'ActivationStartDate' | 'Porting activation start date.' |
| 'ACTIVATION_END_ DATE' | 'ActivationEndDate' | 'Porting activation complete date.' |
| 'AUDIT_TYPE' | 'Audit Type' | 'Audit Type' |
| 'CLASS_FLAG' | 'CLASS Flag' | 'CLASS Flag' |
| 'CNAM_FLAG' | 'CNAM Flag' | 'CNAM Flag' |
| 'FOR_SP_ID' | 'For SP Id' | 'For Service Provider Identification' |
| 'ISVM_FLAG' | 'ISVM Flag' | 'ISVM Flag' |
| 'LIDB_FLAG' | 'LIDB Flag' | 'LIDB Flag' |
| 'RN_FLAG' | 'RN Flag' | 'RN Flag' |
| 'ROUTING_NUMBER_ FLAG' | 'Routing Number Flag' | 'Routing Number Flag' |
| 'WSMSC_FLAG' | 'WSMSC Flag' | 'WSMSC Flag' |
| 'TIMESTAMP' | 'Timestamp' | 'Timestamp' |

| | | |
|---|---|---|
| 'USERNAME' | 'Username' | 'Username' |
| 'NRC_NAME' | 'NRC Name' | 'NRC Name' |
| 'MEDIATOR_SP_ID' | 'Mediator SP Id' | 'Mediator Service Provider Identification' |
| 'NRC_SV_ID' | 'NRC SV Id' | 'NRC Subscription Version Identification' |
| 'SUBSCRIPTION_TYPE' | 'Subscription Type' | 'Subscription Type' |
| 'SV_SMS_ID' | 'SV SMS Id' | 'Subsription Version SMS Identification' |
| 'SV_SOA_ID' | 'SV SOA Id' | 'Subscription Version SOA Identification' |
| 'ACTIVATION_DUE_ DATE' | 'Activation Due Date' | 'Porting Activation Due Date' |
| 'BILLING_ID' | 'Billing Id' | 'Billing Identification' |
| 'BLOCKED_FLAG' | 'Blocked Flag' | 'Blocked Flag' |
| 'CHANGED_BY_SP_ID' | 'Changed By SP Id' | 'Changed by Service Provider Identification' |
| 'CONCURRENCE_FLAG' | 'Concurrence Flag' | 'Concurrence Flag' |
| 'CREATED_BY_SP_ID' | 'Created By SP Id' | 'Created by service provider identification' |
| 'CUSTOMER_TYPE' | 'Customer Type' | 'Customer Type' |
| 'DISCONNECT_DUE_ DATE' | 'Disconnect Due Date' | 'Disconnect Due Date' |
| 'EFFECTIVE_RELEASE_ DUE_DATE' | 'Effective Release Due Date' | 'Effective Release Due Date' |
| 'LOCKED_FLAG' | 'Locked Flag' | 'Locked Flag' |
| 'NEW_SP_ AUTHORIZATION_FLAG' | 'New SP Authorization Flag' | 'Set if Porting Request Authorized by New Service Provider' |
| 'OLD_SP_ AUTHORIZATION_FLAG' | 'Old SP Authorization Flag' | 'Set if porting request authorized by old service provider.' |
| 'PORTING_TIME' | 'Porting Time' | 'Porting Time' |
| 'NUMBER_RETURNED_ DUE_DATE' | 'Number Returned Due Date' | 'Number Returned Due Date' |

| | | |
|---|---|---|
| 'PREORDER_ AUTHORIZATION_CODE' | 'Preorder AuthorizationCode' | 'Preorder Authorization Code e.g. Letter of Agency' |
| 'PREV_STATUS_TYPE_ CODE' | 'Prev Status Type Code' | 'PreviousStatusTypeCode' |
| 'PRE_SPLIT_ SUBSCRIPTION_TN' | 'Pre-Split Subscription TN' | 'Telephone Number prior to a number range split.' |
| 'PTO_FLAG' | 'PTOFlag' | 'Port to OriginalFlag' |
| 'STATUS_CHANGE_ CAUSE_CODE' | 'Status Change Cause Code' | 'StatusChangeCauseCode' |
| 'STATUS_TYPE_CODE' | 'StatusTypeCode' | 'Status Type Code' |
| 'USER_LOCTN_TYPE' | 'UserLoctnType' | 'User Location Type' |
| 'USER_LOCTN_VALUE' | 'UserLoctnValue' | 'User Location Value' |
| 'NEW_STATUS_TYPE_ CODE' | 'New Status Type Code' | 'New StatusTypeCode' |
| 'OLD_STATUS_TYPE_ CODE' | 'Old Status Type Code' | 'Old StatusTypeCode' |
| 'ACTIVE_FLAG' | 'ActiveFlag' | 'Active Flag' |
| 'STATUS_NAME' | 'StatusName' | 'Status Name' |
| 'PHASE_INDICATOR' | 'PhaseIndicator' | 'Phase Indicator' |
| 'DESCRIPTION' | 'Description' | 'Description' |
| 'DISPLAY_NAME' | 'Display Name' | 'Display Name' |
| 'LANGUAGE' | 'Language' | 'Language' |
| 'SOURCE_LANG' | 'SourceLang' | 'Source Language' |
| 'PRODUCT_TYPE_CODE' | 'ProductTypeCode' | 'Product Type Code' |
| 'TIMER_TYPE_CODE' | 'TimerTypeCode' | 'Timer Type Code' |
| 'TIMER_VALUE' | 'Timer Value' | 'Timer Value' |
| 'UNIT_OF_MEASURE' | 'Unitof Measure' | 'Unit of Measure' |
| 'FAILURE_DATE' | 'Failure Date' | 'Failure Date' |
| 'SUBSEQUENT_PORT' | 'Subsequent Port' | 'Is this a subsequent port?' |
| 'SYNC_LABEL' | 'Sync Label' | 'Unique label used for synchronization' |

| | | |
|---|---|---|
| 'SYNC_REQD_FLAG' | 'Sync Required Flag' | 'Flag to indicate if synchronization was required across line items' |
| 'RANGE_COUNT' | 'Range Count' | 'Count of the number of Number Ranges which are being ported together' |
| 'COMMENTS' | 'Comments' | 'User Comments' |
| 'NOTES' | 'Notes' | 'User Notes' |

# C

# Configuring Adapters

## C.1 Adapter Architecture

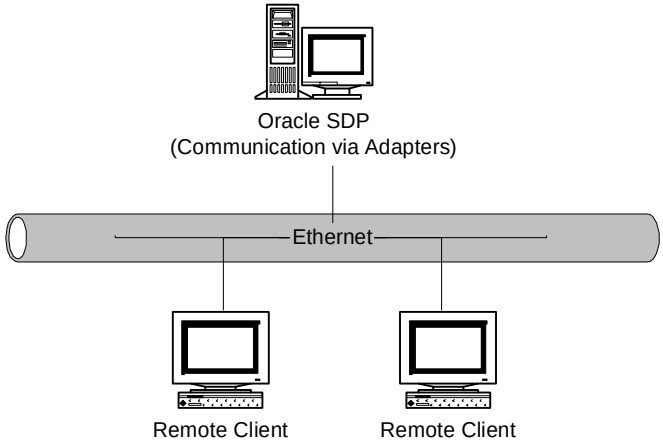
Oracle SDP requires interaction with remote systems & applications to perform application level processing. Remote systems & applications require that any interaction with them happens via well defined interfaces. These interfaces are defined via “Protocol Data Units” (PDUs) for communication over a certain type of network medium.

To encompass a wider range of remote applications, a generic framework is needed that will allow communication to happen via:

- Customized PDUs.
- Independent Network Interfaces.
- Session Level Control.
- Extension of the framework for any specific needs.

The Adapter framework of Oracle SDP is a framework, written in Java language, that is generic enough to be customized for most remote application needs. The framework is made up of components that can be extended, via derivation, for specialized applications and protocols. The Adapter framework represents a “gateway” interface for external communications using Oracle Provisioning

***Figure C–1 Remote Communications via Adapters.***

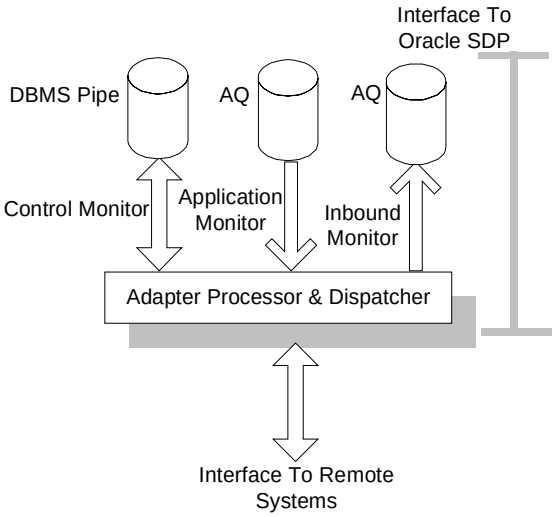


## Architecture

The architecture of Oracle SDP adapters can be described as follows:

- Control Message Monitor
- Application Message Monitor
- Inbound Message Monitor

***Figure C–2 Adapter Processor & Dispatcher***



## C.2 Control Message Monitor

The control message monitor is responsible for monitoring a DBMS pipe for control messages. The adapters understand certain types of control messages for application level control. Some of the control messages are:

- OPEN
- CLOSE
- SHUTDOWN_NORMAL
- SUSPEND
- RESUME
- FTP
- CLOSEFILE

The control messages are initiated from the Oracle SDP internals and acted upon by the adapter. Users can define their own control messages and extend the adapter framework to interpret those new control messages. The name of the DBMS pipe on which the adapter listens for control messages is the “channel name”. The “channel name” is passed as a parameter to the adapter during its invokation. The “channel name” is supposed to be unique. To acknowledge the receipt of control messages, the adapter itself creates a reply DBMS pipe. The name of this pipe is always the channel name along with “_REPLY”. For e.g., if the “channel name” passed to the adapter during its invoke is “ATT_12345” then there will be two DBMS pipes created for “Control Message Monitoring”. The name of one DBMS pipe will be “ATT_12345” which will carry control messages for the adapter. The other DBMS pipe name would be “ATT_12345_REPLY” which will carry acknowledgment messages from the adapter to the processes of Oracle SDP, that run inside the DBMS.

The control message monitor is a java thread in the main adapter processor & dispatcher. The thread checks for messages in the DBMS pipe after a delay. The delay can be fine tuned in the adapter.properties file via property Adapter.ControlMessage delay.

### Application Message Monitor

The application message monitor is responsible for processing application level messages in Oracle SDP. Unlike control message monitor, application message monitor listens on the AQ of Oracle SDP. The AQ, which is responsible for delivering application messages to the application message monitor, is called an outbound AQ. The AQ name on which the application message monitor listens on is declared in adapter.properties via property Adapter.OutboundAQ.name. Application message monitor deques the messages from the outbound AQ using the "adapter name" specified as the consumer of the AQ. This "adapter name" is provided as a parameter when the adapter is launched. While processing of a message, if an application error occurs, the application message is not dequed and an error is sent on the inbound AQ. In case of certain severe error, an error message is sent back and the adapter automatically performs its own shutdown. e.g., FTP failures during the FTP of a file via the FileAdapter.

The application message monitor is a java thread in the main adapter processor & dispatcher. The thread checks for messages in the outbound AQ after a delay. The delay can be fine tuned in the adapter.properties file via property Adapter.ApplicationMessage.delay.

## Inbound Message Monitor

The inbound message monitor is responsible for delivering messages, from remote applications and processed by the adapter, to Oracle SDP. The inbound message monitor writes the messages to the inbound AQ. The name of the inbound AQ is determined by the property Adapter.InboundAQ.name in the adapter.properties file.

The inbound message monitor is a java thread in the main adapter processor & dispatcher. The thread can be provided a message by the application. The provided message is then dispatched to the inbound AQ from where it is collected by Event Manager.

## Adapter Processor & Dispatcher

The main application processing is performed by the "Adapter Processor & Dispatcher". The main functionality of the adapter processor is to monitor all the monitors and dispatch their processing to respective application elements.

The application adapter is an abstract java class that needs can be extended to provide application level functionality. The core components, the monitors, are available in the base class and can be either extended or used by the derived class.

## Control Messages

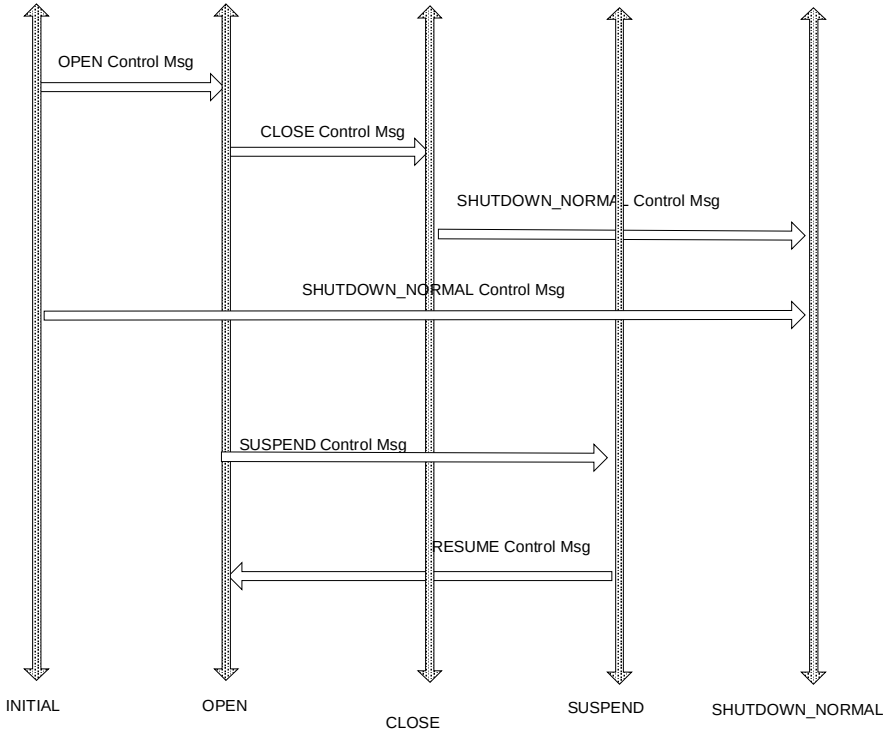
Control messages are messages that are processed by the Control Message Monitor. The various OPERATION types defined for control messages in Oracle SDP adapters is as follows:

| OPERATION | Description |
|---|---|
| OPEN | Changes the state of the adapter to OPEN. In OPEN state, the adapter is ready to process application level messages. |
| CLOSE | Changes the state of the adapter to CLOSE. In CLOSE state, the adapter does not process any application level messages. |
| SHUTDOWN_NORMAL | Changes the state of the adapter to SHUTDOWN. In this mode, the adapter exits as a process with exit code 0. |
| SUSPEND | Changes the state of the adapter to SUSPEND. In this state, the adapter does not process any application level messages. |
| RESUME | Changes the state of the adapter to OPEN from a suspended state. |
| FTP (Specific to the FileAdapter) | Performs the “ftp” operation of the file to a remote server. This control message is an example of user defined control message and has been implemented in the file adapter. |
| CLOSEFILE (Specific to the FileAdapter) | Performs the “close” operation with respect to a file adapter. The “close” operation may require a “footer” to be written to the file being managed by the file adapter. This user defined control message also instructs the FileAdapter to start processing another default file.<br>This control message is another example of an user defined control message and has been implemented in the file adapter. |

# C.3 State Machine

The “base adapter framework” provides a base abstract “Adapter” class. This class can be extended from, via derivation in Java, and made “concrete” by overriding the method “performApplicationMessageProcessing”. Control messages dictate the state of the adapter at any given point. Following are the states in the current “adapter framework”:

***Figure C–3 Adapter States***



### FE Attributes

Fulfillment Element Attributes are required by an adapter implementation in order to connect to the remote element and perform its function. An example is the IP address for a file adapter. The IP address is required to connect to the remote machine and FTP a file. A Fulfillment element can implement several Adapter technologies like File and TCP/IP. All the defined FE attributes will passed to the adapter upon opening the adapter for communication. The FE attributes are passed as XML elements in the OPEN message. Please refer to the SDP reference manual on configuring FE attributes and their values.

### Connect Procedures

The connect procedure is executed by the SDP Fulfillment Element Controller after spawning a new adapter. By default the connect procedure will call the API *xnp_adapter.open()*, which can be over ridden by a user defined procedure. The open API retrieves all the FE attributes, constructs a OPEN message and sends it to the adapter. The adapter can use the attributes to implement the functionality. Please refer to the SDP reference manual for more information on the connect and disconnect procedures.

### Disconnect Procedures

The disconnect procedure is executed by the SDP Fulfillment Element Controller before shutting down an adapter. By default the disconnect procedure will call the API *xnp_adapter.close()*, which can be over ridden by a user defined procedure. The close API send a CLOSE message to the adapter. The close for the specific implementation can close the connection with the remote element.

### Framework Java Classes

Java classes in the adapter framework are organized in package "oracle.apps.xnp.adapter". There are three main categories which further classify the framework classes:

- core - package "oracle.apps.xnp.adapter.core"
- db - package "oracle.apps.xnp.adapter.db"
- impl - package "oracle.apps.xnp.adapter.impl"

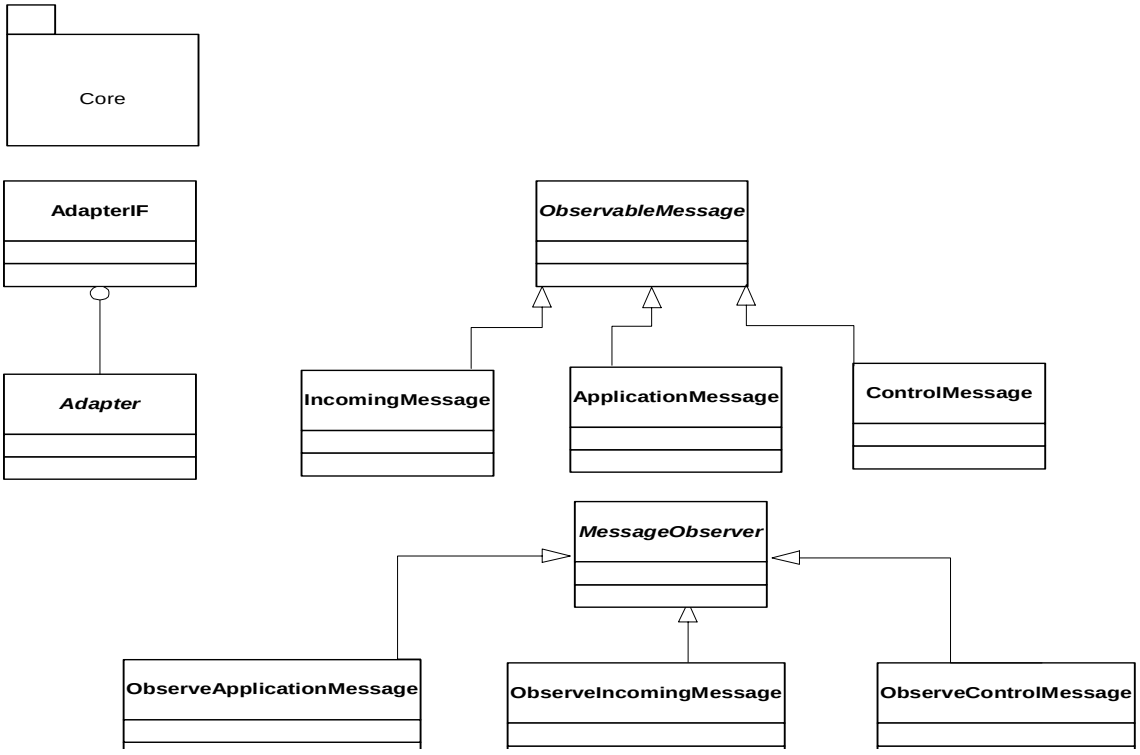
The "core" package defines all the Java classes responsible for the core framework. The classes of the "core" package can be "subclassed" to provide application functionality.

The "db" package defines all the Java classes responsible for any database interaction. The "DBConnection" and "DBAccess" classes, in the "db" package, are the primitives responsible for performing the database connection and performing any database access. There are also classes in this package responsible for "pipe" I/O and "AQ" I/O.

The "impl" package is the actual implemented adapters. This package has the "file" sub-package which includes the "FileAdapter". Along with the "FileAdapter", in the sub-package "file", are the classes for "FTPClient" and "ScanDirectory".

## C.4 Class Hierarchy

The class hierarchy of the Java classes that are in the “core” package is as follows:



### Java Core Package

Following Java classes make up the “core” package:

- AdapterIF - Interface class for any adapter framework
- Adapter - abstract class, implements AdapterIF
- ObservableMessage - abstract class, extends Observable, implements Runnable
- MessageObserver - abstract class, implements Observer, has ObservableMessage
- IncomingMessage - concrete class, extends ObservableMessage and implements the run() method
- ApplicationMessage - concrete class, extends ObservableMessage and implements the run() method
- ControlMessage - concrete class, extends ObservableMessage and implements the run() method
- ObserveApplicationMessage - concrete class, extends the MessageObserver and implements the update() method
- ObserveIncomingMessage - concrete class, extends the MessageObserver and implements the update() method
- ObserveControlMessage - concrete class, extends the MessageObserver and implements the update() method
- MessageParser - singleton class, has the primitives to parse an XML message
- FileManager - singleton class, has all the primitives to perform file level operations clustered together

## Adapter Class

The “Adapter” class, in the package “core” , is the base class for all adapter implementations. Overriding implementations should extend this class and implement the “abstract” methods. Overriding implementations can also override default “public” & “protected” interfaces of this class.

The following “public” methods are available as interfaces of the Adapter class:

- Public Methods

| Return Type | Method Name | Description |
|---|---|---|
| boolean | Open(Object aMessage, Hashtable aParsedEntries) | This method is invoked when a control message of type “OPEN” is received by the adapter. The input parameter aMessage is of type MessageIF and aParsedEntries is the key / value pairs of the aMessage in a parsed format. This method represents the OPEN state of the adapter. The default behavior of this method, as implemented, is to return “true”.<br>This method is invoked by the “handleControlMessage” method. If the implementation returns a “true” value then everything is considered normal. Implementation class can override this method to perform any initial network connections or initialization. |
| boolean | Close(Object aMessage, Hashtable aParsedEntries) | This method is invoked when a control message of type “CLOSE” is received by the adapter. The input parameter aMessage is of type MessageIF and aParsedEntries is the key / value pairs of the aMessage in a parsed format.<br>This method is invoked by the handleControlMessage method. If the implementation returns a “true” value then everything is considered normal.<br>The implementation class can override this method to perform any close on network connections or files. |

| | | |
|---|---|---|
| boolean | Resume(Object aMessage, Hashtable aParsedEntries) | This method is invoked when a control message of type “RESUME” is received by the adapter. The input parameter aMessage is of type MessageIF and aParsedEntries is the key/value pairs of the aMessage in a parsed format.<br>This method is invoked by the handleControlMessage method. If the implementation returns a “true” value then everything is considered normal.<br>The implementation class can override this method to perform any re-opening of network connections which might have been closed when a “SUSPEND” message had arrived. |
| boolean | Suspend(Object aMessage, Hashtable aParsedEntries) | This method is invoked when a control message of type “SUSPEND” is received by the adapter. The input parameter aMessage is of type MessageIF and aParsedEntries is the key/value pairs of the aMessage in a parsed format.<br>This method is invoked by the handleControlMessage method. If the implementation returns a “true” value then everything is considered normal.<br>The implementation class can override this method to perform any closing of network connections upon the receipt of this message. |

| | | |
|---|---|---|
| boolean | ShutdownNormal(Object aMessage, Hashtable aParsedEntries) | This method is invoked when a control message of type “SHUTDOWN” is received by the adapter. The input parameter aMessage is of type MessageIF and aParsedEntries is the key/value pairs of the aMessage in a parsed format.<br>This method is invoked by the handleControlMessage method. If the implementation returns a “true” value then everything is considered normal.<br>The implementation class can override this method to perform any cleanup operations before the adapter is shutdown. |
| void | handleControlMessage(Object aMessage) | This is the method that is invoked upon receipt of any “Control message”. Implementation class can override this method to provide a state machine for control messages. |
| | | |
| boolean | initialize(String aArguments[]) | This method initializes data of the base adapter. The arguments to this method are the same argument array provided to the “main” of the implementation. If you do override this method in the implementation class, then make sure to invoke the super.initialize with appropriate arguments. |

## File Adapter Class

File Adapter provides an implementation of the “adapter framework”. To provide an implementation of the “adapter framework”, the implementation class has to extend the base “Adapter” class, from the “core” package, and override the method “performApplicationMessageProcessing”. Along with that, the implementation has a choice of several methods it can override or extend to provide application level functionality.

The following public & protected member methods have been either extended or provided by FileAdapter.java:

- Public Methods

| Return Type | Method Name | Description |
|---|---|---|
| boolean | Open(Object aMessage, Hashtable aParsedEntries) | The input parameter aMessage is of type MessageIF and aParsedEntries is the key / value pairs of the aMessage in a parsed format. This method represents the OPEN state of the adapter. This method belongs to the base adapter class and has been overridden in the FileAdapter. The reason this method has been overridden is because of the following:<br>It invokes createFileName method to create the name of a file if FILE_ NAME is not provided.<br>It invokes createHeader to create the header that needs to be written in a newly created file. |
| boolean | performControlMessageProcessing(Object aMessage, Hashtable aParsedEntries) | This method belongs to the base adapter class and has been overridden in the FileAdapter. This method is invoked for user defined “control” messages. The user defined “control” messages are FTP and CLOSEFILE in the case of FileAdapter. |
| boolean | performApplicationMessageProcessing(Object aMessage) | This is the ONLY “abstract” method in the base adapter class. In the case of FileAdapter, the messages, of type MessageIF, are written to a file that can be FTPed. |

| | | |
|---|---|---|
| boolean | performScanDirectory(Object aMessage, Hashtable aEntries) | Will invoke the "initializeScanDirectory" to initiate a scanning thread. This scanning thread is what's called the "Inbound Message Monitor" |
| void | initializeScanDirectory() | In case of file adapter, instantiates a new ScanDirectory class and assigns it to "itsScanDirectory" a protected member of FileAdapter class and of type ScanDirectory. |

- Protected Methods

| Return Type | Method Name | Description |
|---|---|---|
| String | getAbsoluteFileNameBasedOn HomeDir(String aFileName) | Will append the HOMEDIR (FE Attribute) to the filename and return the new file name. |

### FE Attributes for a File Adapter

The following FE attributes have to be defined for the file adapter.

| Attribute Name | Remarks |
|---|---|
| HOMEDIR | The home directory is used to hold the file being constructed. |
| IP_ADDRESS | The IP address of the remote machine to which the file will be uploaded. |
| PORT | The port on which the remote FTP server is running. Standard port is 21. |
| USER_ID | The remote user on the remote machine. |
| PASSWORD | Password for the user. |
| REMOTEDIR | The remote directory is the directory on the remote machine to which the newly constructed file will be uploaded. |
| OUT_ARCHIVE_DIR | The directory in which the file will be archived after the FTP. |
| SCANDIR | The directory from which files will be read and sent to SDP for processing. |
| IN_ARCHIVE_DIR | The directory to which the file will be archived after the adapter reads it. |
| | |

### Extending the File Adapter

The File Adapter can be extended to provide user level functionality.

#### Source for MyFileAdapter.java:

package examples.file;

```
import java.util.Hashtable;
import oracle.apps.xmp.adapter.impl.file.ScanDirectory;
import oracle.apps.xmp.adapter.impl.file.FileAdapter;
import oracle.apps.xmp.adapter.db.MessageIF;
import oracle.apps.xmp.adapter.core.Adapter;

import examples.File.MyScanDirectory;

public class MyFileAdapter extends FileAdapter {
```

```
public MyFileAdapter()
{
super();
}

public StringBuffer createFooter()
{
StringBuffer aBuffer = new StringBuffer();
aBuffer.append("/* TRAILER FOR FTP (C) ORACLE CORPORATION */\n");
return(aBuffer);
}

public StringBuffer createHeader()
{
StringBuffer aBuffer = new StringBuffer();
aBuffer.append("/* START OF HEADER */\n");
aBuffer.append("/* HEADER FOR FTP (C) ORACLE CORPORATION */\n");
aBuffer.append("/* CONTENTS OF THIS FILE ARE ENCRYPTED */\n");
aBuffer.append("/* END OF HEADER */\n");
return(aBuffer);
}
```

(Continued on next page)

(Continued from previous page)

```
public String createFileName()
{
/* Should return an absolute file name and not a relativefile name. e.g.,
 absolute file name: /export/home/user1/MyData */

return(super.createFileName());
}

public String performMessageTransformation(Object aMessage)
{
/* The input argument, aMessage, is always of type
   MessageIF.

   RULES:

   a. Return a transformed string after you have transformed the message.
   b. The transformed message gets written to the file.
   c. If you don't want this message to go to the file, just return(null);
```

```
*/

MessageIF aNewMessage = (MessageIF)aMessage;
return(aNewMessage.toString());
}(Continued on next page)

 (Continued from previous page)


public void initializeScanDirectory()
{
/* Specialize my own scanner */

itsScanDirectory = new MyScanDirectory(itsProperties);
}

public static void main(String [] aArgs)
{
Adapter aAdapter = new MyFileAdapter();

if (aAdapter.initialize(aArgs))
{
if (aAdapter.invokeControl())
{
if (aAdapter.invokeApplication())
{
System.err.println("Adapter launched!");
}
else
{
System.err.println("Failed to invoke application!");
}
}
else
{
System.err.println("Failed to invoke application!");
}
}
else
{
 (Continued on next page)

 (Continued from previous page)
System.err.print("Wrong number of arguments: ");
for (int i = 0; i < aArgs.length; i++)
```

```
{
if (i > 0)
{
System.err.print(" ");
}

System.err.print(aArgs[i]);
}
System.err.println();
}
}
};
Source for MyScanDirectory.java:

package examples.file;

import java.lang.String;
import java.util.Properties;

import oracle.apps.xnp.adapter.impl.file.ScanDirectory;

public class MyScanDirectory extends ScanDirectory {

public MyScanDirectory(Properties aProperties)
{
super(aProperties);
}

public String performMessageTransformation(
String aFileName, int aMessageNumber, String aMessage)
{
/* aFileName - Is the file name from which this message has
   been extracted.
   aMessageNumber - Is the index of the message. Begins at 0.
   aMessage - The message retrieved.

   If you return a null from this method, then the message
   is not sent at all on the inbound Q. So to ignore header
   and trailer messages, just return null.
*/

return(super.performMessageTransformation(aFileName,
aMessageNumber,aMessage));
}
};
```

## Extending Adapters

To extend the base FileAdapter, the following need to be performed:

- Derive from the class FileAdapter. e.g., public class MyFileAdapter extends FileAdapter
- In the constructor of MyFileAdapter make sure super() is invoked.
- The following methods can be overridden:

| Return Type | Method Name | Description |
|---|---|---|
| Script Adapter | | |
| StringBuffer | createFooter() | The footer that can be written to a file when a CLOSEFILE operation is performed. |
| StringBuffer | createHeader() | The header that can be written to a file when the first OPEN operation on the adapter is performed. |
| String | createFileName() | Create the name of the file as desired. This function is invoked when a default file needs to be created. |
| String | performMessageTransformation(Object aMessage) | If an application message needs to be transformed, then override this method to provide transformation of choice. |
| void | initializeScanDirectory() | If you write your own scanner, the thread responsible for “Inbound Message Monitor”, then instantiate the scanner by overriding this method. The scanner instantiation should be assigned to “itsScanDirectory” member. |

The Script Adapter has been provided here in the form of source code as an example of how to write new adapters.

### Source for ScriptAdapter.java:

package oracle.apps.xnp.adapter.impl.script;

```
import java.lang.String;
import java.lang.Runtime;
import java.lang.Process;
import java.lang.Exception;
import java.util.Hashtable;
import java.io.InputStream;
import java.io.IOException;

import oracle.apps.xnp.adapter.core.Adapter;
import oracle.apps.xnp.adapter.db.TraceLog;

public class ScriptAdapter extends Adapter {

public ScriptAdapter()
{
super();
}

public boolean performApplicationMessageProcessing(Object aMessage)
{
return(true);
}

public boolean performControlMessageProcessing(
Object aMessage, Hashtable aParsedEntries)
{
String anOperation =
(String)aParsedEntries.get("OPERATION");
```

(Continued on next page)

(Continued from previous page)

```
Runtime aRuntime = Runtime.getRuntime();

try {
Process aProcess = aRuntime.exec(anOperation);
InputStream anInput = aProcess.getInputStream();
int aRead = 0;

byte aByte[] = new byte[4096];

TraceLog.write(itsUniqueID,"ScriptAdapter:performControlMessageProcessing:
```

```
 Output from the OPERATION: " + anOperation,TraceLog.INFORMATIONAL);

while(true)
{
try {
aRead = anInput.read(aByte);
}

catch(IOException anException) {
break;
}

if (aRead == -1)
{
/* EOF */
break;
}
else
{
String aString = new String(aByte,0,aByte.length);
TraceLog.write(itsUniqueID,"ScriptAdapter:performControlMes
```

(Continued on next page)

(Continued from previous page)

```
 sageProcessing: " + aString,TraceLog.INFORMATIONAL);
}
}

aProcess.waitFor();
}

catch(Exception anException) {

TraceLog.write(itsUniqueID,"ScriptAdapter:performControlMessageProcessing: An
 exception was generated: " + anException.toString(),TraceLog.INFORMATIONAL);
}

return(true);
}

public static void main(String [] aArgs)
{
Adapter aAdapter = new ScriptAdapter();
```

```
if (aAdapter.initialize(aArgs))
{
if (aAdapter.invokeControl())
{
System.err.println("Adapter launched!");
}
else
{
System.err.println("Failed to invoke application!");
}
}
else
nued from previous page)

System.err.print("Wrong number of arguments: ");
for (int i = 0; i < aArgs.length; i++)
{
if (i > 0)
{
System.err.print(" ");
}

System.err.print(aArgs[i]);
}

System.err.println();
}
}
};
```

### FE Attributes for a Script Adapter

The following FE attributes have to be defined for the file adapter.

| Attribute Name | Remarks |
|---|---|
| IP_ADDRESS | The IP address of the remote machine on which the command is to be executed. |
| PORT | The port on which the telnet server is running. Standard port is 23. |
| USER_ID | The remote user on the remote machine. |
| PASSWORD | Password for the user. |
| | |

## C.5 Writing New Adapters

Writing new adapters using the Adapter Framework is straightforward. To start with, study the Script & File Adapter source code and follow the guidelines mentioned in this section.

Following is a step-wise wizard that will allow the development of new adapters using the adapter framework:

- Derive from the "Adapter" class. e.g., public class MyAdapter **extends** Adapter
- Override the method "**performApplicationMessageProcessing**"
- Override any other method from the base Adapter class as needed
- Write a "main", just like the one, as mentioned in the source of Script & File Adapter
- Make sure the CLASSPATH is set to pick up the respective "jar" & "zip" files
- Make sure the "adapter.properties" file has been configured with the right property values
- Invoke the adapter and check for traces in the trace file name

## C.6 Frequently Asked Questions

Q. What does a connect procedure look like?

A. The following is an example of a connect procedure:

(Service->Fulfillment Elements->SW Versions->Details - has the

connect procedure (Add/Edit procedure)

```
xnp_adapter.open(fe_name, channel_name, sdp_internal_err_code, sdp_
internal_err_str);
```

Q. What does a disconnect procedure look like?

A. The following is an example of a disconnect procedure:

(Service->Fulfillment Elements->SW Versions->Details - has the

disconnect procedure (Add/Edit procedure)

```
xnp_adapter.close(fe_name, channel_name, sdp_internal_err_code, sdp_internal_
err_str);
```

Q. Do I need the connect / disconnect procedures?

A. Yes. You do need the connect and disconnect procedure as they perform specialized functionality.

Q. How are the messages from the outbound Q dequeued?

A. The messages are dequeued from the outbound Q using the adapter name. The adapter name is used to queue the messages in the outbound Queue. The place to configure this in the GUI is: Administration->Connection Management Utility. Select the Fulfillment Elements (on the left drop list) and click on the tab Adapters. Make sure that there is only *one adapter* per FE. Usually, provide the adapter name same as that of the FE name.

Q. When designing a message in iMessageStudio, I selected the "Test Message" tab and then selected the (list of values) Consumer List. From the consumer list, I picked the FE name which will process the message. But it seems that the messages (messages in the outbound Q) are dequeued based on the adapter name. How does this co-relation take place?

A. When the message is queued, the "adapter name" provided for the FE is extracted and used for queuing. Make sure that you choose the right FE name for which you have configured an adapter name.

Q. How can I override base adapter methods and write my own adapter?

A. Perform the following:

- Derive the new adapter class from the base adapter class.

- Override methods in the derived class which implement your functionality.
- Compile your class. So for e.g., if your class is named MyClass and is part of the package mycom.adapter then the absolute path name for your class would be mycom.adapter.MyClass.

Now configure your adapter using the GUI's Fulfillment Element.

- (Service->Fulfillment Element). You have to define a new attribute for the adapter
- (Service->Fulfillment Element->SW Version->Attributes) named **IMPLEMENTATION_CLASS**. The value of this attribute will be the exact path name of your implemented class. i.e., **IMPLEMENTATION_CLASS** mycom.adapter.MyClass

Q. What is the name of the log file generated by Java based adapters?

A. The name of the file has a prefix of "AdapterLog". The rest of the name is dependent on the channel name.

Q. How is the class IncomingMessage implemented or extended? How can I specify my own method of receiving messages and put them in the inbound queue?

A. The Incoming Message needs to be implemented as follows:

1. Derive a class from IncomingMessage class:

```
class MyIncomingMessage extends IncomingMessage {
};
```

2. Use the constructor of MyIncomingMessage has Properties as a single argument:

```
class MyIncomingMessage extends IncomingMessage {
public MyIncomingMessage(Properties aProperties)
{
}
};
```

3. Provide a method "performAction" that either returns a message of type MessageIF or a Vector of MessageIF:

```
public Object performAction()
{
a. Perform your processing.
b. If error, set:
   1. itsError(type boolean inherited from IncomingMessage) to
  "true". (itsError = true)
   2. itsErrorDescription(type String inherited from
  IncomingMessage) to a descriptive description. e.g.
  itsErrorDescription = "I cannot process something";
   3. return(null);
c. If everything is normal then
   return(either a single MessageIF or a Vector of MessageIF);
}
```

4. Provide a method "performPostAction" that returns either a "true" or "false". The purpose of this method is that if you want to perform some post processing after "performAction" then you can do it here. Otherwise, no need to provide this method. In most cases, you'll not be needing this method.

5. Perform the following in you class that has the main adapter code: i.e.,

```
class MyAdapter extends Adapter {
 };
```

   a. Override the method Open:

```
class MyAdapter extends Adapter {

protected MyIncomingMessage itsMyIncomingMessage = null;

public boolean Open(Object aMessage, Hashtable aEntries)
{
1. Extract any FE attributes you need for sanity checks
   from aEntries.
2. Instantiate & initialize the MyIncomingMessage class:
   e.g.,

   itsMyIncomingMessage =
new MyIncomingMessage(itsProperties);
// itsProperties is inherited from Adater class

   itsMyIncomingMessage.initialize()
   {
  itsMyIncomingMessage.setFEAttributes(something);
```

```
// if you want to set any attributes then
// write a method named setFEAttributes for
// class MyIncomingMessage and invoke them here.

  itsIncomingMessage = itsMyIncomingMessage;
// itsIncomingMessage is inherited from Adapter
// class. Perform this assignment.

  itsIncomingMessageMonitor =
new ObserveIncomingMessage(this,
itsIncomingMessage);
// itsIncomingMessageMonitor is inherited from
// Adapter. The above statement initializes and
// starts the IncomingMessage thread so that it
// polls the "performAction" method defined in
// class MyIncomingMessage.
   }

return(true);
}
};
```

The main points for an Incoming Message thread are:

- After the "itsIncomingMessageMonitor" variable is instantiated, the "IncomingMessage" thread is launched. (the "run" method of the thread is now in an infinite loop with a delay that can be customized).
- The "IncomingMessage" thread invokes the "performAction" method as defined in MyIncomingMessage from the "run" loop. Based on what's return and error conditions, it either puts the returned contents in the Inbound AQ or shuts the adapter down because of error.

Q Can the method performApplicationMessageProcessing of class "FileAdapter be overloaded?

A. Yes, the method performApplicationMessageProcessing can be overloaded. However, the method has a "synchronized" specifier for its access. (In the class FileAdapter). So if you do plan to override this method, make sure you make it "synchronized" as well.