Oracle® SDP Number Portability

Implementation Guide

Release 11i

August 2000

Part No. A86289-01

ORACLE®

Oracle SDP Number Portability, Realease11i

Part No.A86289-01

Contents

Appendix A: Public APIs

Appendix B: Sample Workflows

Appendix C: SDP Parameters

Appendix D: Number Portability Views

Appendix E: Configuring Adapters

Appendix F: Implementing Number Portability Process Synchronization

Send Us Your Comments

Oracle SDP Number Portability, Release 11i

Part No. A86289-01

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 Number Portability, Release 11i.

This Detailed Implementation Guide provides information and instructions about the implementation of the Oracle SDP Number Portability 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
- Typical Release Dependencies
- Setting up Profile Options
- Setting up System Profiles
- Considerations for Future Upgrades
- Appendices
  - Public APIs
  - Sample Workflows
  - SDP Parameters
  - Number Portability Views
  - Configuring Adapters
  - Implementing Number Portability Process Synchronization

Related Documents

The following are resources related to Oracle Number Portability:

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 Number Portability User's Guide*
- *Oracle Provisioning User's Guide*
- *Developing Number Portability Applications Reference Guide*
- *Oracle Number Portability Technical Reference Manual*
- *Developing XML Message Based Application Reference Guide*
- *Con figuration "How To's"*
  - User Defined Workflows
  - Workflow Fulfillment Actions
  - Timers
- *Aim Documentation*
  - CR010: Scope, Objectives, and Approach
  - BR020: Business Requirements Mapping Form
  - BR110: Application Setup
  - TE070: Unit Test Scripts

Conventions

The following conventions are also used in this manual:

Convention	Meaning
. . .	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 Number Portability

Considerations for Planning an Implementation Project

The following items are part of planning an Oracle Number Portability Implementation:

- Overview of Oracle Number Portability
- Application architecture
- Modules
- Features and functions
- User roles
- Other considerations

Number Portability Defined

Oracle Number Portability enables users of telecommunication services on one switching system to move or "port" numbers to a different switching system.

Oracle Number Portability provides the capability for consumers to keep their telephone numbers when:

- Switching between telecommunication service providers
- Moving from one physical location to another

Changing one telecommunication service for another telecommunication service NP is tightly integrated with Oracle Provisioning. In addition, ONP is capable of interfacing with a service provider's existing provisioning system or other commercial service provisioning products. ONP is installed using Oracle's three tier Internet Computing architecture and is fully web-enabled.

NP Development

Number Portability Applications can be defined using the Oracle Service Delivery Platform and the predefined NP Work Items, Work Item parameters, Workflows and Core Functions. This guide explains all the predefined Work Items, Workflow activities, NP Core Library Functions and provides an insight into defining new Work Items, Workflow Activities and other building blocks for developing a Number Portability application.

Routing Numbers

A routing number makes the porting of numbers between switching systems possible. The routing number is mapped to the new telecommunication switching system on the one end and linked to the customer's current dialing number on the other end. Each service provider allocates certain numbers as routing numbers for number portability.

Every service provider using Oracle Number Portability has a routing number repository for this purpose. The following table lists the locations where routing numbers are recorded and the purpose of each location.

Routing Number Repositories

Use	Location
Execution	The service provider's local database
Control	Central databases maintained by national or regional number centers. These are known as NRCs in the United States. They are also known as NPAC (United States), S-NPAC (Sweden), and CRDC (Belgium). This system is generically referred to as NRC in Oracle's Number Portability product.

This table lists:

1. Where routing numbers are stored
2. When numbers are used in the porting process

Types of Number Portability

The table following displays the three commonly accepted types of number portability.

The Different Types of Number Portability

Type	Description
Service provider portability	The ability to retain the same telephone numbers as the user changes from one service provider to another. For example, Dr. Jon Smith moves local telephone service for his office from Euro Telecom to the new emerging carrier SuperTel. The physical location of Dr. Smith's office has not changed.
Location portability	The ability to retain existing telecommunication numbers without impairment of quality, reliability, or convenience when moving from one physical location to another. For example, Dr. Jon Smith moves his office from Brussels to Gothemburg and retains his existing telecommunications numbers with SuperTel.
Service portability	The ability to retain existing telecommunication numbers without impairment of quality, reliability, or convenience when switching from one telecommunications service to another telecommunication service provided by the same service provider. For example, Dr. Jon Smith moves a subset of the telecommunications numbers for his office from wireline service to wireless service with SuperTel. When signing up for wireless service, he retains the existing telecommunications numbers with the same set of features that he had with wireline service.

Understanding Orders

There are a number of order activities that you carry out in Oracle Number Portability. The following table lists the types of order activity that you can perform in the application and provides a short description of each.

Examples of Order Activity in Oracle Number Portability

Order Type	Description
Port In	A port in request is usually initiated at the recipient service provider side by a new customer requesting that you provide a port in capability.
Port Out	A port out request can occur in one of two ways: ▫ It is triggered internally at the donor service provider side based on the same customer contact as at the recipient side. ▫ It is possible for a customer to contact the donor service provider directly in order to initiate a port out request to another service provider.

Note: Standard activities are provided for the user to build business processes for each order type. These order types do not come seeded with the product.

Ways to Submit a Porting Request

There are three ways to submit a porting request to Oracle Number Portability. The table following describes each.

Ways to Submit a Porting Request

Submission Type	Description
You use the application graphical interface.	Information is entered through the Port In Request window, accessed via the Test Center.
You use the APIs provided with the application. (See Appendix A: Public APIs)	This include the following APIs: ▪ Process Order API ▪ Submit Order ▪ Cancel Order ▪ DRC Process Order

Porting Phases in Oracle Number Portability

There are four porting phases in Oracle Number Portability. The following table provides a description of each.

Porting Phases in Oracle Number Portability

Phase	Description
Inquiry	A customer calls the new service provider and inquires as to the feasibility of porting his/her number. For example, Dr. John Smith may call three new service providers to ask what the rates would be if he ported his number. Note that it is possible for multiple porting inquiry requests to be placed for the same telephone number at the same time.
Ordering	A customer calls and requests service from a new service provider and asks to keep his/her telephone number from the old service provider. Only a single porting order request can be placed for a telephone number at any given time. For example, Dr. John Smith can only select one service provider to port his number.
Active	A customer's telephone number is currently active in the network with the new service provider. Note that the Active phase should not be confused with the Active flag. The Active flag is defined for each porting status to indicate whether the porting status is a valid value that can be used by the application.
Old	A porting request that has been canceled or disconnected, etc. It is suggested that, after a specified time period, porting requests with a status in the Old phase should be archived or purged.

Typically, the phase of the service order in the number porting process most often determines the tasks that you perform with a service order (but not always).

For example:

- You perform the tasks associated with entering orders in the Ordering phase of number porting.
- You perform monitoring and number range maintenance and their component tasks in any phase of a number porting process.

Porting Statuses in Oracle Number Portability

Porting statuses are used to track the progress of a porting request.

A service order may have more than one status throughout its life cycle. You can define a new status through Oracle Number Portability for either of the following reasons:

- To better reflect the terminology and business processes of your organization
- To understand the progression of a porting request

Once a status is defined it can be referenced in Workflow and process logic. These are the places where manipulation of the statuses occur.

Key Components of a Porting Status Definition

In Oracle Number Portability, each porting status is user-definable. It must also be associated with one of the pre-defined phases of a porting request. User-defined porting statuses provides flexibility to users (for example, naming conventions are different for each country). Associating a porting status with a pre-defined porting phase allows applications to categorize porting statuses, adding to the manageability of porting requests.

The porting phases are:

- Inquiry
- Ordering

Service Bureau Installation

Number Portability 11i supports hosting for multiple service providers in a single installation site because all the number portability data is indexed using the service provider and each user in the site can only belong to one service provider. Also every transaction in the service bureau mode takes the service provider into account.

Users can be assigned to a specific Service Provider from the configuration screens.

The value of the 'SP_NAME' profile option is used as a default for the local service provider. All the number portability data is indexed using the service provider and the site can offer number portability services to multiple Service Providers at the same time.

Equal Access Configuration

Oracle Number Portability 11i can also be configured to support Equal Access functionality through the use of a subscription type field in the port-in and port-out operations.

Subscription Type	Meaning
NP	Implies Local Number Portability
EA_LOCAL	Implies porting from one long distance carrier to another long distance carrier for inter-LATA calls only. A good example of Inter-LATA call is a call from San Francisco to San Jose. Inter-LATA calls can be viewed as calls within the same state.
EA_LD	Implies porting from one long distance carrier to another for inter state calls. An example of an inter-state call is a call from Chicago to New York.
EA_INTL	Implies porting from one long distance carrier to another for international calls.

The section below describes how the fields in the subscription version entity will be interpreted in case of Equal Access. Only the attributes having meaning in the Equal Access area is mentioned

Field Name	Description
SV_SOA_ID	A unique sequential number
OBJECT_REFERENCE	A unique identifier for this equal access subscription.
SUBSCRIPTION_TN	The telephone number to be ported from one long distance carrier to another.
SUBSCRIPTION_TYPE	Indicates if the subscription is of Equal Access type.
DONOR_SP_ID	The long distance carrier losing the customer.
RECIPIENT_SP_ID	The long distance carrier gaining the customer.
STATUS_TYPE_CODE	Status of this subscription. Refer to configuring the status type codes.
CREATED_BY_SP_ID	The service provider who initiated the porting.
MEDIATOR_SP_ID	Mediating Service Provider.

Field Name	Description
CHANGED_BY_SP_ID	N/A
ROUTING_NUMBER_ID	N/A
PREV_STATUS_TYPE_ CODE	Saved status type code.
ACTIVATION_DUE_ DATE	The date on which the subscription will be effective.
NEW_SP_DUE_DATE	The date the recipient long distance carrier wants the subscription activated.
OLD_SP_CUTOFF_DUE_ DATE	The date the donor long distance carrier wants the telephone number to be deactivated.
EFFECTIVE_RELEASE_ DUE_DATE	The date the long distance service from the donor will be disconnected.
STATUS_CHANGE_ DATE	The date the status last changed.
CREATED_DATE	The date the subscription was created.
MODIFIED_DATE	The date the subscription was modified.
CONCURRENCE_FLAG	Indicates concurrence from the other carrier.
NEW_SP_ AUTHORIZATION_FLAG	Indicates if recipient long distance carrier has authorized the porting.
OLD_SP_ AUTHORIZATION_FLAG	Indicates if the donor long distance carrier has authorized the porting.
STATUS_CHANGE_ CAUSE_CODE	The reason the status was changed.
CUSTOMER_ID	A unique identifier for the customer.
CUSTOMER_NAME	Name of the customer.
CUSTOMER_TYPE	N/A
CONTACT_NAME	N/A
ADDRESS_LINE1	Address of the customer.
ADDRESS_LINE2	Address of the customer.
CITY	City
STATE	State

Field Name	Description
ZIP_CODE	Zip Code
COUNTRY	Country of the customer
PHONE	Contact phone of the customer.
MOBILE	Mobile number of the customer.
FAX	Tax number of the customer.
PAGER	Pager
PAGER_PIN	Pager pin.
EMAIL	Email
INTERNET_ADDRESS	N/A
PRICE_CODE	N/A
PRICE_PER_CALL	N/A
PRICE_PER_MINUTE	N/A

Refer to the user guide for configuring the service provider, geographic areas and other number portability data.

Routing Interconnect Areas

For Equal Access, you can also specify the geographic areas valid for each routing number. This can allow for querying up which Routing Number to assign to a customer based on the area in which they live. This has to be done as part of a FE locator procedure for an FP.

SP Gateway

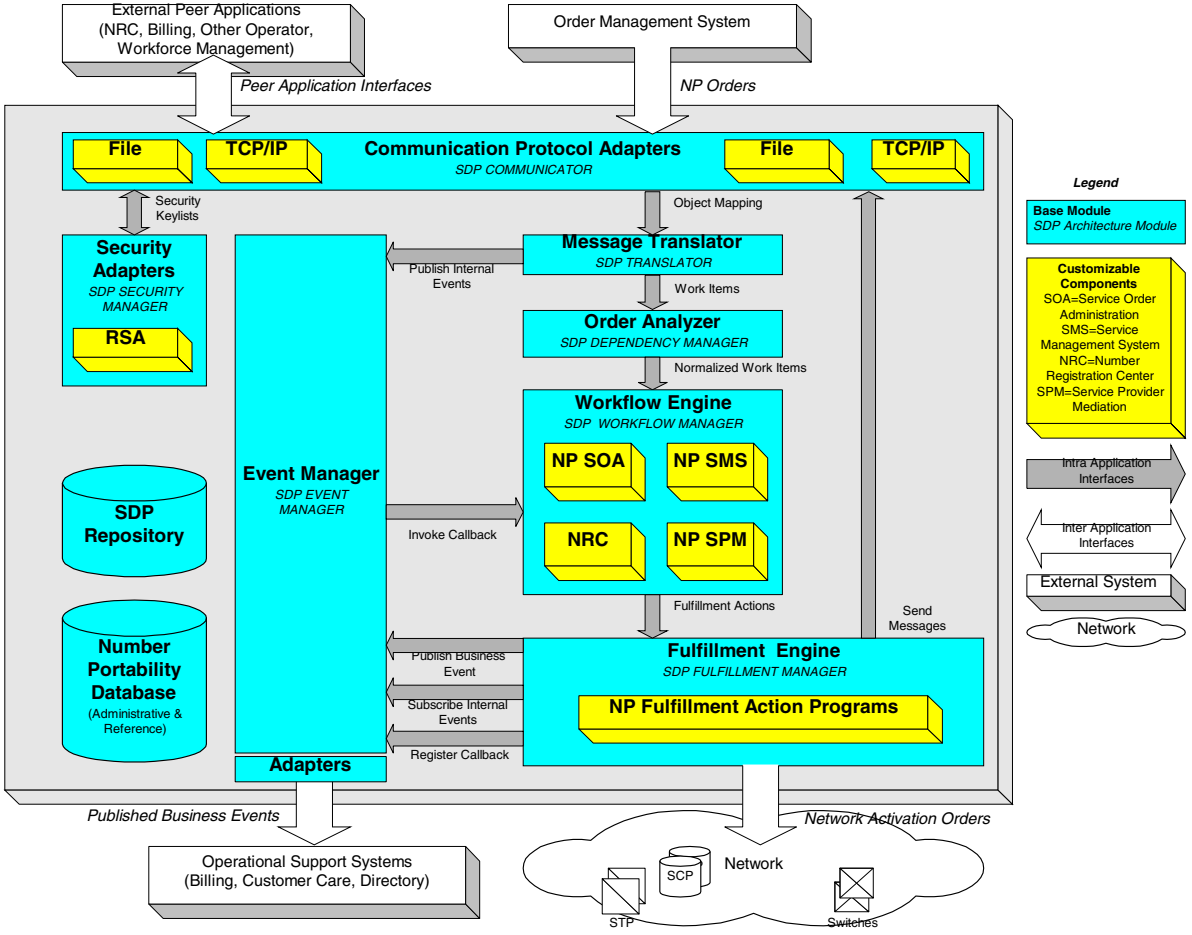
You would need to associate an FE (of type SP Gateway) with every Service Provider defined. This is how the Workflow Send procedure can determine which adapter to send message to. For the NRC, create a SP of type NRC and associate an FE (of type NRC Gateway).

Steps for NP Application Development

1. Setup Service Providers and Fulfillment Elements for each of the external interface (Gateways).
2. Setup Services or Work Item parameters.
3. Setup Application Messages, message responses and group responses
4. Define SDP Fulfillment Procedures for the pre-configured Fulfillment Actions.
5. Load lookups for messages, FAs, statuses, phases on to workflow lookups
6. Define Number Portability Business Process Workflows.
7. Define the processing logic, validation logic and default processing logic for the messages.
8. Associate workitem parameters based on the workflow activities, incoming and outgoing messages/events.
9. Run Orders

Note: See the Provisioning guide for defining and associating Fulfillment Procedures, Fulfillment Actions and Work Items.

See Appendix D: Number Portability Views.


External Peer Applications
(NRC, Billing, Other Operator,
Workforce Management)
Order Management System
Peer Application Interfaces
NP Orders
File
TCP/IP
Communication Protocol Adapters
SDP COMMUNICATOR
File
TCP/IP
Security
Keylists
Object Mapping
Security
Adapters
SDP SECURITY
MANAGER
RSA
Publish Internal
Events
Message Translator
SDP TRANSLATOR
Work Items
Order Analyzer
SDP DEPENDENCY MANAGER
Normalized Work Items
Workflow Engine
SDP WORKFLOW MANAGER
NP SOA
NP SMS
NRC
NP SPM
Event Manager
SDP EVENT
MANAGER
SDP
Repository
Invoke Callback
Fulfillment Actions
Send
Messages
Number
Portability
Database
(Administrative &
Reference)
Publish Business
Event
Subscribe Internal
Events
Register Callback
Fulfillment Engine
SDP FULFILLMENT MANAGER
NP Fulfillment Action Programs
Adapters
Published Business Events
Network Activation Orders
Operational Support Systems
(Billing, Customer Care, Directory)
SCP
Network
STP
Switches
Legend
Base Module
SDP Architecture Module
Customizable
Components
SOA=Service Order
Administration
SMS=Service
Management System
NRC=Number
Registration Center
SPM=Service Provider
Mediation
Intra Application
Interfaces
Inter Application
Interfaces
External System
Network

Application Architecture

Oracle Number Portability Scope

Service provider portability is the scope of this release of ONP. ONP enables users of telecommunication services to keep their telephone numbers when they move between telecommunication service providers.

Oracle Number Portability Architecture

Oracle Service Delivery Platform (SDP) addresses the challenges of the global number portability initiative with the following functions:

- Service Management System (SMS)
- Service Order Administration (SOA)
- Service Bureau (SB)

The complete SDP platform is designed to support number portability and will satisfy the donor, recipient and participant service providers' requirements.

The SDP platform can also be customized to support number portability requirements for a central system.

Modules

The following are modules for Oracle Number Portability:

Communication Protocol Adapter

An incoming XML message will be taken by the appropriate Communication Protocol Adapter. For the inbound XML message, the adapter puts the message in the inbound queue for the Event Manager.

For outbound messages, the application puts the message in the outbound queue. The appropriate communications protocol adapter retrieves the message and sends it to the associated fulfillment element.

Note: If the incoming message is not in XML, customized adapters are required to convert the message into XML.

If the outgoing needs to be converted from XML to another format, customized adapters are required to convert the message to a different format.

Message Translator

The Translator module converts incoming messages from XML to the required ONP format. An XML parser could be used in the incoming processing logic of the message.

The Translator module is not used for outgoing messages.

Order Analyzer

See Oracle Provisioning for information about the order analyzer and its ability to manage dependencies within an order.

Workflow Engine

This module specifies the actual flow of actions (known as Fulfillment Actions) that need to be executed in order to satisfy an application functionality. In essence, this is the module that could re-use the Fulfillment Actions to customize any new functionality such as NP Service Provider Mediation or the NRC itself. The Workflow Engine would determine the Fulfillment Actions to execute for each Work Item and determine the Fulfillment Elements that it would need to talk to. It would then place this information on an outbound queue for the Fulfillment Engine and notify the Event Manager of its interest in the outcome of the execution. Once it gets an event notification of the outcome of the execution of the Fulfillment Action (by the Event Manager), the engine would proceed to complete the Work Item and pick the next Work Item in the queue for the given order.

Fulfillment Engine

The Fulfillment Engine determines the Fulfillment Elements on which the Fulfillment Actions must be applied. There are two key methods by which a Fulfillment Action can be applied on Fulfillment Elements. Both methods can co-exist within a single user-defined Workflow.

In one method, the Fulfillment Engine determines which Fulfillment Procedure to execute on the Fulfillment Elements. The Fulfillment Procedure would be a PL/SQL program written by the end-user administrator during application setup to provide the functionality for the system being built. This method is typically used by Oracle Number Portability to provision network elements.

In the second method, the Fulfillment Engine applies pre-defined functions on Fulfillment Elements without requiring user-defined PL/SQL programs. The functionality to apply these pre-defined functions on Fulfillment Elements is pre-built in the application. The pre-built functions include SendMessage, Receive Message and Publish Event which are commonly used to support messaging in Number Portability.

Once the execution of the Fulfillment Action is completed, it would notify the Event Manager of its completion. This asynchronous architecture allows the Fulfillment Engine to execute programs which may reside in another system, be a pre-defined Oracle Workflow or which may not be PL/SQL based. All outbound messages would be sent directly from the Fulfillment Engine API to the Communication Protocol Adapters for peer communications. Inbound messages from peer systems would use the Event Manager and log the incoming messages as Events.

Event Manager

The Event Manager is a generic Publish-Subscribe module which registers interest of various subscribers to different event types. The subscribers could be the Translator (in which case the event gets propagated as a new order), Workflow Engine (in which case the event restarts a Workflow which is waiting on an external event) or the Fulfillment Engine (in which case a Fulfillment Procedure can be invoked to complete a previous action). The Event Manager can also be used by any module to publish a business event which could be used to notify external systems to internal ONP events. An example would be that an M-EVENT-REPORT message from NPAC for a new customer port-in could publish a business event NEW_CUSTOMER_ALERT which could be used by external Order Management Systems to create a port-in request for the new customer and Network Planning systems to plan for assigning a line and telephone connection to the new customer. The Oracle Provisioning repository Configuration module would be used to define or modify business events and assign the subscribers to the event type. This module would also use the publish-subscribe features of Advanced Queues available in Oracle.

SDP Repository

The core SDP repository allows the user to create orders and configure fulfillment elements. The Configuration module within the core repository allows the user to setup the system after installation. An example would be Work Item, Fulfillment Action, Fulfillment Procedure and Fulfillment Element definitions.

ONP Database

The ONP database would contain entities for storing ONP specific data such as Subscription Version and Service Provider. The user will also be able to use the SDP Repository Configuration module to customize the ONP specific information. It would take in the appropriate site configurations and update the repository with customized database objects which are optimized for runtime performance. On completion of the message definition, the Configuration would compile definition stored in the database and create a set of stored procedures to implement the SendMessage and ReceiveMessage commands.

ONP Service Order Administration

The main function of Number Portability Service Order Administration (NP-SOA) is to mediate orders between the service provider and the NRC. Once the number portability order has been mediated by the NP SPM, the NP SOA supplements the incoming order information with the necessary ONP data elements and converts it into a Subscription Version. The information is sent for processing by the NRC, which too would confirm receipt. On the due date, the NP SOA would send a confirmation notification for the number porting which would trigger the NRC to broadcast the porting information.

Usage

NP-SOA includes any communication between service providers through the central system. This can include:

- Donor Initiation Phase
- Recipient Initiation Phase
- Central System Initiation Phase

Example: Porting Order request by a customer.

Note: The Porting Order request is typically generated for the following:

- Recipient Service Provider by an order from an upstream system.
- Central System by a message received from the recipient service provider.
- Donor Service Provider by a message received from the central system.

ONP Service Management System functions

Number Portability Management System (NP SMS) coordinates ported number activation, change and disconnect orders from the NRC with the network elements via the SDP Provisioning Component (PROV) to provision ONP services for individual customers.

Upon receipt of an incoming ONP Subscription Version notification from the NRC, NP SMS validates the data and translates the routing information to provisioning messages that are sent to the domain-level NEMS for provisioning the appropriate network elements. NP SMS is also responsible for querying the local database in response to audit requests from the NRC or SOA and sends replies containing ONP data for the records requested.

Usage

NP-SMS includes any communication with service providers which require network activation. This can include:

- Donor Activation Phase
- Recipient Activation Phase
- Central System Activation Phase
- Participant Activation Phase

Example: Porting Activation request by the recipient operator on or after the due date.

Note: A Porting Activation Request is typically generated for one of the following:

- Recipient Service Provider by a business event from an external system.
- Central System by a message received from the recipient service provider on or after the due date.
- Donor Service Provider by a message received from the central system.
- Participant Service Provider by a broadcast received from the central system.

Features

The main features of ONP are as follows:

- Standard open APIs for integration with upstream ordering systems
- Tight integration with Oracle Provisioning to support provisioning needs
- Flexible GUI-based Workflow accommodating customizations of service providers' customized business rules using a library of ONP functions
- Automatic code generation for sending and receiving predefined messages to and from external systems in the industry-standard XML format
- Web-enabled interface using Oracle's standard look-and-feel
- Local database that maintains a copy of number portability data and porting requests
- Flexible Java-based adapter architecture to support interfaces to external systems
- Monitoring tools available to track porting order requests and messaging.

User Roles

Finding Information in the NP Center

You use the NP Center to view various kinds of information about orders within the Oracle Number Portability system. This include the following:

- Information about an existing order
- Information about network elements associated with an order

The information displayed in the NP Center is read-only, and can not be modified in any way.

Viewing Order Information in the NP Center

To view information about an order, perform the following steps.

> **Note:** The information displayed in the NP Center is read-only.

Prerequisites

The order must exist in the system before you can view information about it.

Steps

1. In the Navigator, choose **Operations > NP Center**.
1. Select the Orders tab.
2. Select an order from the Subscriptions list at the left.

   The Summary screen for that order displays. See Guidelines following for a description of the form fields and their meaning.
3. Click **Details** to display information about a service provider associated with this order.
4. Select the other tabs at the bottom of the screen to display additional information, if desired.
5. Click **Notifications** to open the Notifications Inbox, if desired.

   See Managing Notifications for details of the Notifications Inbox, if necessary.
6. Click the **X** in the upper right-hand corner of the screen to close it, or select Close Form from the File menu.

Guidelines

The table following lists the kinds of information that is available from the Orders tab of the NP Center.

Information Available from the Orders Tab

Tab	Description
Summary	▪ Subscription Summary Customer name and telephone number, order status, routing number, etc. ▪ Recipient Name and access code of the new service provider. This is Recipient in case of Number Porting, or the new long distance provider in case of Equal Access. ▪ Donor Name and access code of the old service provider. This is Donor in case of Number Portability or the old long distance provider in case of Equal Access. ▪ Mediator Name and access code of mediating service provider. This is Number Registration Center in case of Number Portability, or Local Service Provider in case of eqUal Access.
Features	▪ The Features tab is not available unless special features have been defined for this order. To enable features, set profile option ENABLE_FEATURES to Y (Yes).
Transaction Log	▪ Event Type Type of event that affected the porting record. For example, this could be Status Change or Modification. ▪ Event Actual event, for example, the status of PENDING. ▪ Time Stamp Time when the event took place.

Information Available from the Orders Tab

Tab	Description
Work Items	▪ Order ID Service Delivery Platform internal order identification number. ▪ Order Number External order number as entered in the Order Entry System. ▪ Work Item Work item number assigned by the system for this order. ▪ Status State of this work item. It can be received, running, ready, pending, or complete.
Failed Downloads	▪ Service Provider Code Code for the Service provider who failed to download the broadcast information ▪ Service Provider Name Service provider name ▪ Failure Date Date when the failure took place ▪ Final Failure Date Date when total failure was declared, meaning that there were no more retries ▪ Attempted Retries Number of retries attempted to download the information successfully

Information Available from the Orders Tab

Tab	Description
Others	▪ Subscription Owner Service provider code for the owner of the subscription record (in case of Service Bureau) ▪ Pre-Order Authorization Number which gives the authorization to the service provider to switch the subscriber to another carrier. ▪ Activation Due Date Date when activation must take place ▪ Status Change Date Date when the status was last updated ▪ Invoice Due Date Date when the SP must be billed ▪ Disconnect Due Date Date when subscriber wishes to disconnect service ▪ Number Returned Due Date Date when the new Service Provider returns the number to the old Service Provider after disconnecting service ▪ Effective Release Due Date Date when the subscriber is completely removed from the old SP system ▪ Donor Cutoff Due Date Date when the donor wants to remove the subscriber from the system

Network Information

To view information about the network elements associated with an order, perform the following steps.

> **Note:** The information displayed in the NP Center is read-only, and can not be modified in any way.

Prerequisites

The order must exist in the system before you can view information about it.

Steps

1. In the Navigator, choose **Operations > NP Center**.
1. Select the Network tab.
2. Select an order from the Subscriptions list at the left.

   The Summary screen for that order displays. See Guidelines following for a description of the form fields and their meaning.
3. Click **Details** to display information about a service provider associated with this order.
4. Select the other tabs at the bottom of the screen to display additional information, if desired.
5. Click **Notifications** to open the Notifications Inbox, if desired.

   See Managing Notifications for details of the Notifications Inbox, if necessary.
6. Click the **X** in the upper right-hand corner of the screen to close it, or select Close Form from the File menu.

Guidelines

The table following lists the kinds of information that is available from the Networks tab of the NP Center.

Information Available from the Network Tab

Tab	Description
Summary	▪ Subscription summary This includes the customer telephone number, routing Id, etc. ▪ Routing service provider name Service provider name of the recipient provider (owning the routing number) ▪ Routing service provider code Service provider code ▪ Mediating Number Registration Center name Name of the mediating service provider ▪ Mediating Number Registration Center code Service provider code

Information Available from the Network Tab

Tab	Description
Features	▪ The Features tab will not be available unless special features have been defined for this order. To enable features, set profile option ENABLE_FEATURES to Y (Yes).
Work Items	▪ Order Id Order identification number ▪ Order Number External order number as entered in the Order Entry System ▪ Work Item Work item number assigned by the system for this order. ▪ Status State of this work item. (This could be received, running, ready, pending, or complete.)
Provisioning Map	▪ Features Type Indicates the feature type that has been provisioned on the network element.For example, this could be: Routing Number, LIDB, or CNAM. (If the Features functionality is not enabled, then only the Routing number for Number Portability is provisioned. This is also called the Primary Routing Number.) ▪ Network Element Network element name on which the provisioning for this porting record has taken place. ▪ Network Element Type Network element type on which the provisioning for this porting record has taken place. ▪ Provision Status Status of the provisioning of the network element.

Events Diagnostics

You use the Callback Event Diagnostics page to monitor the runtime activities of the callback registrations used by the Event Manager to route messages to the correct recipients.

Prerequisites

None

Steps

1. In the Navigator, choose **Diagnostics > Event**.

   The Callback Event Diagnostics page opens.

2. Enter a value for any of the displayed fields, then click **Find**.

   For example:

   - To search on the order ID, enter a range of values in the Order ID fields and click **Find**.
   - To search on order status, select one from the drop-down list, then click **Find**.
   - To search on a message type, sent to a given Service Provider, enter the Service Provider Code in the Receiver field and click **Find**.

   Any diagnostic messages that meet the search criteria display under Event Details.

3. Close the browser window to exit Diagnostics.

Timer Diagnostics

You use the **Timer Diagnostics** page to search for timer messages based on known details. Timers are used by the system for enforcing time-based rules. This could be, for example, jeopardy or service level agreement restrictions.

Prerequisites

None

Steps

1. In the Navigator, choose **Diagnostics > Timer**.

   The Timer Diagnostics page opens.

2. Enter a value for any of the displayed fields, then click **Find**.

   For example:

   - To search on the order Id, enter a range of values in the Order Id fields, and click **Find**.
   - To search on a specific Timer type, click **List** to open a list of the valid types, select one, then click **Find**.
   - To search on active timers, enter ACTIVE in the status field and click **Find**.

   Any diagnostic messages that meet the search criteria display under Message Details.

3. Close the browser window to exit this diagnostics page.

Using the Configuration Diagnostics Tools

You use the Configuration Diagnostics tool to validate the existing system configuration. The validation process consists of two levels:

Error: This type indicates the items that must be corrected for the system to work as expected.

Warnings: This type indicates potential problems, but which in some case, are acceptable.

> **Note:** It is recommended that you run this utility after any major change in the system configuration.

Prerequisites

None

Steps

1. In the Navigator, choose **Diagnostics > Configuration**.

   The Configuration Diagnostics page opens. This action dynamically executes a validation tool that returns information on the validity of the system configuration.

2. Close the browser window to exit this diagnostics page.

Managing Geographic Areas

You perform a number of tasks with geographic areas. These include:

- Defining a new geographic area type
- Defining a new geographic area

Using the iMessage Studio

You use the iMessage Studio to manage the task of creating new messages. In creating a message, you must perform the following tasks in the order listed.

Step	Task	See
1.	Define the details of the message.	Creating a New Message
2.	Add elements to the message. This includes text, number and date/time fields.	Adding Message Elements
3.	Set the structure of the relationships between the elements of the message.	Setting the Message Structure
4.	Define the source of the information contained in the message.	Defining the Message Data Source
5.	Create any special logic for processing this message.	Defining Your Own Message Processing Logic
6.	Compile the message.	Compiling a Message
7.	Test the message.	Sending a Test Message

If desired, and it meets your business needs, you can also create a custom notification message.

Creating a New Message

To create a new message, perform the following steps.

> **Warning: Do not create message bodies greater than 32 Kb in length.**

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 message.
3. Enter a name for the message.
4. Select the Details tab.
5. Select Message from the drop-down list of available types.
6. Enter a short name for the message in the Display Name field.

   This name is referenced by the message studio when generating procedures.
7. Enter a brief description of the message.
8. Select a priority for the message.

   This value sets the priority for the message in the outbound or inbound message queue.
9. Select a queue name from the drop-down list.

   If you are uncertain of which to chose, then keep the default value unchanged.
10. Chose a user Responsibility for this message from the drop-down list.
11. Enter the DTD Location path.

    This value sets the path structure to the file that holds all the schema (Document Type Definitions) for this message. The file is named <message>.dtd, where <message> corresponds to the name of the message you are currently defining.
12. Close the window.

    You are prompted to save your changes.

Refer to Adding Message Elements for details of how to enter the information for each message element that makes up this message.

Adding Message Elements

When creating a message definition, the Message Code and MESSAGE are automatically shown as mandatory messages elements. The message code is defined as the root element.

Warning: You must never delete the root element, the message code for this message.

To add a message element to a message, perform the following steps.

Prerequisites

You must create the message details first, before you can add message elements to it. See Creating a New Message for details.

Steps

1. In the Navigator, choose **Setup > Message Definition > iMessage Studio**.
2. Select the message to which you wish to add elements from the Message Codes list on the left.
3. Select the Elements tab.
4. Enter a name for the element that you are adding.
5. Enter the datatype for your message element.

   The message element datatype is the XML tag that appears in the message for this element. Supported datatypes are:

   - Text
   - Number
   - Date and time
6. Enter the maximum allowed length of the data.

7. Enter a default value for this message element.
   - You only specify default values for message elements that are parameters.
   - The Send and Publish generated procedures raise an error at run-time if an element does not have a value.
8. Check Mandatory if the message element must be included with the message.
9. Check Parameter if this element is to be used as an argument in generated procedures.

   If a message element is marked as a parameter, then the default value is used. The iMessage Studio generates a CREATE_MSG(), SEND_MSG(), and a PUBLISH_MSG() procedures with the element as a parameter defaulted to the value specified.
10. Enter a value for the sequence order that you wish the parameter to appear in all the generated procedures.
11. Close the window.

    You are prompted to save your changes.

Setting the Message Structure

The iMessage Studio provides a hierarchical diagramming that you use to define the message structure and to set relationships between message elements. Only predefined elements can be part of this hierarchy.

You can format an XML message by defining its structure here. This allows for complex messages, with master detail relationships.

> **Note:** Elements can be defined more than once within the structure. Also, message elements can be used multiple times throughout the message.

To define the message structure, perform the following steps.

Prerequisites

You must create the message details first, before you can set the message structure. See Creating a New Message for details.

Steps

1. In the Navigator, choose **Setup > Message Definition > iMessage Studio**.
2. Select the Structure tab.

   Use the Structure tab to set up the message element hierarchy you require for this message. The onscreen Usage Notes describe how to modify the hierarchy.
3. Close the window.

   You are prompted to save your changes.

Using the iMessage Diagnostics Tool

You use the **iMessage Diagnostics** page to search for runtime messages and events based on known details.

Prerequisites

None

Steps

1. In the Navigator, choose **Diagnostics > Message**.

   The iMessage Diagnostics page opens.
2. Enter a value for any of the displayed fields, then click **Find**.

   For example:

   - To search for failed messages, enter FAILED in the status field and click **Find**.
   - To search for messages sent to a given Service Provider, enter the Service Provider Code in the Receiver field and click **Find**.
   - To search on Code and Event Indicators, click **List** to open a list of the valid indicators, select one, then click **Find**.

   Any diagnostic messages that meet the search criteria display under Message Details.

3. Click the Message Id to view the details for a message returned by the search.
4. Click the XML link to view the actual XML version of the message (if your browser supports this option).
5. Close the browser window to exit this diagnostics page.

Note: If the current message has a status of FAILED, then Message Details page displays a description of the error. A **Fix Message!** link also appears.

Clicking this link re-enqueues a message on the internal events queue for processing.

Although it is possible to correct the error through this link, it is recommend that correct the error through the Notifications Inbox.

Other Planning Considerations

During ONP installation, profile options and the oracle stack must be considered. ONP supports many profile options. The SP_NAME is discussed below.

Profile Options

The mode of operation as Service Bureau or single Service Provider is enabled upon installation of ONP 3i. The value of the SP_NAME profile option is used as a default for deciding the local default Service Provider. If this profile option is not set, then every user of the system needs to be associated explicitly with a service provider. The user on login, will then be associated with the service provider and any transactions performed by the user will then be attributed to that service provider.

If the profile option SP_NAME is set with the code of a service provider, then it is optional to associate a user with a service provider as the profile option value is taken as the default. This option may be utilized mainly when the installation for a single service provider where specifying the SP for each user is not necessary.

This option is important to consider since porting data is secured using the service provider information and the site can offer number portability services to multiple Service Providers at the same time (the Service Bureau mode).

Ask the regulatory body in your country

- What are the national and regional regulations regarding ONP?
- What is the mandate for delivery of number portability in your country?
- Which types of Number Portability is the regulatory body mandating (e.g., Service Provider Portability only or other types as well)? Is the mandate different for each type of Number Portability?
- Are there one or more central systems for ONP in your country or location?
- If there are multiple central systems, what is the coverage area for each?

Ask your company

- What is your business process flow as a (Service Provider):?
  - Donor,
  - Recipient
  - Participant
- What is the input data required for each business process flow?
- What are the valid actions that can be performed on the ONP service?
- What is the state diagram for a porting request?
- What types of users will receive notifications from the application?
- Are there predefined values such as timer duration, reject reason codes, etc.?
- Are there network elements that require provisioning?
- What are the number ranges your Service Provider serves?
- Who are the other service providers in your area?
- What are the number ranges served by each service provider in your area?
- What are the external interfaces to the application?
- What is the communications mechanism for each interface?
- What messages are passed on each interface? What is the message format?
- Are there any unsolicited messages?
- What events must be published by the application? Who are the subscribers

Typical Release Dependencies

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

- Common Data Model
- Common Schema
- Technical Stack

Oracle Provisioning must be implemented and installed for Oracle number Portability to work.

Setting Profile Options

The following items must be defined:

Work Items in Oracle Number Portability

Oracle Number Portability makes use of workflows to automate business processes in the same manner as does Oracle Provisioning. Work items perform the business and network functions necessary to fulfill a service order request.

However, in the application each work item typically maps to a single workflow, due to the complexity of the business process. That is why workflows are referred to as work items in Oracle Number Portability.

- The workflows delivered with the application were created using the Oracle Workflow Builder. You need to use this application to customize the workflow to meet your business needs.
- Work items comprise function activities. You may need to modify some or all of these while customizing the work items for your business.

The Oracle Workflow installation comes with a standard set of pre-defined notifications and function activities to use in building business processes. The application provides an additional set of pre-defined stub processes, notifications and function activities to support messaging and the porting process. The stub processes must be customized using Oracle Workflow Builder.

Purpose of Work Items

Number Portability 11i provides a predefined set of SDP Work Items which can be used to build a number portability application. Please refer to the Number Portability user guide and SDP user guide for more information on Work Items, Fulfillment Actions and Fulfillment Procedures. The following section describes each of these Work Items and the context in which they can be used for building a Number Portability application. It also provides the mapping of the Work Item to a Workflow.

Every NP Work Item invokes a Workflow to execute the functionality specific to it. To do this, it requires a set of parameters called the Work Item parameters. The Number Portability business process is message or event driven and is between two different service providers or different components of the same service provider. In order to maintain state or context of the business process, each Work Item or Workflow should have a set of standard Work Item parameters in addition to the business specific parameters. The developer has to explicitly assign them to the Work Item.

The next section describes all the predefined Work Items in the Number Portability product. All the Workflows associated with Work Items are part of the **NP Processes** item type. Example Workflows implement the Swedish requirements. ***The Workflow display and internal names should match that of the Work Item display and internal names.***

Predefined Work Items in Oracle Number Portability

Work items are organized into groups known as item types. The table following lists the standard item types delivered with Oracle Number Portability. Do **not** modify any of the standard Oracle Number Portability item types, except NP Processes.

Standard Work Item Types in Oracle Number Portability

Item Type	Internal Name	Description	Customizable
Standard	WFSTD	Contains standard function activities that are commonly used to create business processes.	No
SDP Standard	XDPWFSTD	Contains function activities that are commonly used across both the Oracle Provisioning and Oracle Number Portability applications. It includes, for example, the following functions: Send Message Subscribe to Acknowledgments	No
SDP Lookup Code	XDPCODES	Contains lookup types used for easy selection in workflow activities. This could be, for example, a list of fulfillment actions, or a list of messages.	No
NP Standard	XNPWFSTD	Contains function activities that are specific to Oracle Number Portability. Use these activities in the implementation of your business processes. Examples of standard functions are the following: ▪ Creating a porting order ▪ Determining if porting is possible	No
NP Lookup Code	XNPTYPES	Contains lookup types used for easy selection in workflow activities. This could be, for example, a list of fulfillment actions, or a list of messages.	No

Standard Work Item Types in Oracle Number Portability

Item Type	Internal Name	Description	Customizable
NP Processes	XNPWFACT	Contains country-specific number portability function activities that you customize for different implementations of the application within the same country. It includes, for example, the following processes: ▫ Process porting inquiry ▫ Process porting order ▫ Provision network element	Yes

Guidelines

1. In general, you should use the SDP Standard and NP Standard function activities whenever possible, and only customize the Number Portability activities when absolutely necessary.
2. SDP Lookups and NP Lookups must be updated as you start to configure Service Delivery Platform. Run the Lookups Loader API to update these item types.

Reference

Consult the *Developing Oracle Number Portability Applications Reference Guide*, for details on these item types.

Standard Work Item Parameters

Business processes in Oracle Number Portability are driven by messages or events. Messages can be exchanged between two different service providers or between different components of the same service provider.

- To maintain the state or context of the business process, each work item or workflow must include the set of standard work item parameters in addition to any business specific parameters. The developer must explicitly assign standard work item parameters to the work item.
- Work item parameters act as a parameter pool for all the defined work items in Oracle Number Portability. You define new work item parameters using the existing parameters in the pool. In addition, you can add parameters to the pool to define new work item parameters, or to extend existing work item parameters assigned to a work item.

Mandatory Work Item Parameters

The standard work item parameters listed in the table following are **mandatory** for any Oracle Number Portability work item. All standard parameters are required for any message header that is generated, except the **Messaged** parameter. The Message_ID is used to identify the most recent incoming or outgoing message for a work item instance. This is essential when managing messages (for example, as in re-sending a message).

Note: Oracle Number Portability Work Item Parameters are shown in the application as the Display Name. Sometimes the display name differs from the parameter name. For example, **SP_Name** displays in the application as **Owning Service Provider**.

Mandatory Work Item Parameters

Parameter Name	Description
SP_NAME	Code of the service provider for which the work item is executing. Display Name: Owning Service Provider
OPP_REFERENCE_ID	The reference identifier used on the peer system. All responses to peer requests should populate this field from the REFERENCE_ID field of the request message. Display Name: Opp Reference ID
SENDER_NAME	Code of the sending service provider. This is included in all messages sent from the local system to the peer system. Display Name: Sender Name
RECIPIENT_NAME	Code of the receiving service provider. This is included in all messages sent from the local system to the peer system. Display Name: Recipient Name
MESSSAGE_ID	Identifier of the newly constructed message. The message text is retrievable using the identifier. Display Name: Message ID
STARTING_NUMBER	Starting telephone number in a Service Delivery Platform order. Display Name: Starting Number
ENDING_NUMBER	Ending telephone number in an Service Deliver Platform order. Display Name: Ending Number

Default Work Items in Oracle Number Portability

Note: These work items are optional. You chose whether or not to load the default work items during application installation.

The table following provides a summary of the seeded work items provided with Oracle Number Portability. These work items need to be customized to meet the needs of your installation of the application.applications.

Work Items Seeded by Oracle Number Portability

Work Item	Internal Name	Description
Cancel Disconnect Porting Request from Order Entry	CANCEL_DISC_PORT_REQ_FROM_OMS	Customer contacts current operator to cancel a disconnect subscription request.
Cancel Modify Porting Request from Order Entry	CANCEL_MODIFY_PORT_FROM_OMS.	Customer contacts new operator to cancel a modification previously submitted on a porting request.
Cancel Porting Request from Order Entry	CANCEL_PORT_REQ_FROM_OMS	Customer contacts new operator to cancel a porting request.
Charge New Operator for Porting Request	REC_RECEIVE_CHARGING_NOTIF.	Donor Operator charges recipient operator for the port-out transaction.
Create or Modify Ported Number(s)	PROVISION_PORTED_NUMBER.	Create or Update network elements with new porting data received from Number Registration Center.
Delete Ported Number(s)	DELETE_PORTED_NUMBERS.	Remove porting data from network elements provisioned earlier. This may be in response to a broadcast received from Number Registration Center to carry out this activity.
Disconnect Porting Request from Order Entry	DISC_PORT_REQ_FROM_OMS	Customer contacts current operator to disconnect subscription.
Hold Porting Request from Order Entry	HOLD_PORT_REQ_FROM_OMS.	Customer contacts new operator to place a porting request on hold.
Hold Porting Request from other Operator	HOLD_PORT_REQ_FROM_OPERATOR.	Hold Porting Request received from other operator.
Inquire Donor Operator for Porting Out	PORTING_INQUIRY_FROM_OPERATOR.	Donor operator determines whether porting inquiry request should be approved or rejected.

Work Items Seeded by Oracle Number Portability

Work Item	Internal Name	Description
Inquire Recipient Operator for Porting In	PORTING_INQUIRY_FROM_OMS	New Operator receives porting inquiry from customer care.
Load, Disaster Recovery & Backup of Local Database	LOAD_DISASTER_RECOVERY_BACKUP	Synchronize local database with Number Registration Center.
Modify Porting Request from Operator	MODIFY_PORT_REQ_FROM_ OPERATOR.	Modifies the Porting Request on receiving such a request from the other operator.
Modify Porting Request from Order Entry	MODIFY_PORT_REQ_FROM_OMS.	Customer contacts new operator to modify a porting request.
Number Range Split Request from Order Entry	NUMBER_RANGE_SPLIT	Number Range Split declared by a regulatory board.
Porting Notification Concurrence	RECEIVE_CONCURRENCE	Donor Operator and Recipient Operator receive concurrence notification from Number Registration Center for a porting transaction.
Porting Order initiated by Recipient Operator	PORTING_ORDER_FROM_OMS	Recipient Operator receives porting order from Customer Care.
Query Porting Data	QUERY_REFERENCE_DATA.	Query Number Registration Center for porting data.
Reject Porting Request	PORTING_NOTIFICATION_REJECTION.	Donor Operator and Recipient Operator receive rejection notification from Number Registration Center for a porting transaction.
Remind Operator for Porting Response	PORTING_NOTIFICATION_REMINDER.	Number Registration Center reminds operator that a response is required to proceed with the porting process.
Respond to Porting Order received by Donor Operator	PORTING_ORDER_FROM_OPERATOR.	Donor Operator responds to a Porting Request.
Transfer Number Range Holder from Order Entry	TRANSFER_NUMBER_RANGE_HOLDER	Transfer Number Range Holder declared by a regulatory board.

Work Item Parameters

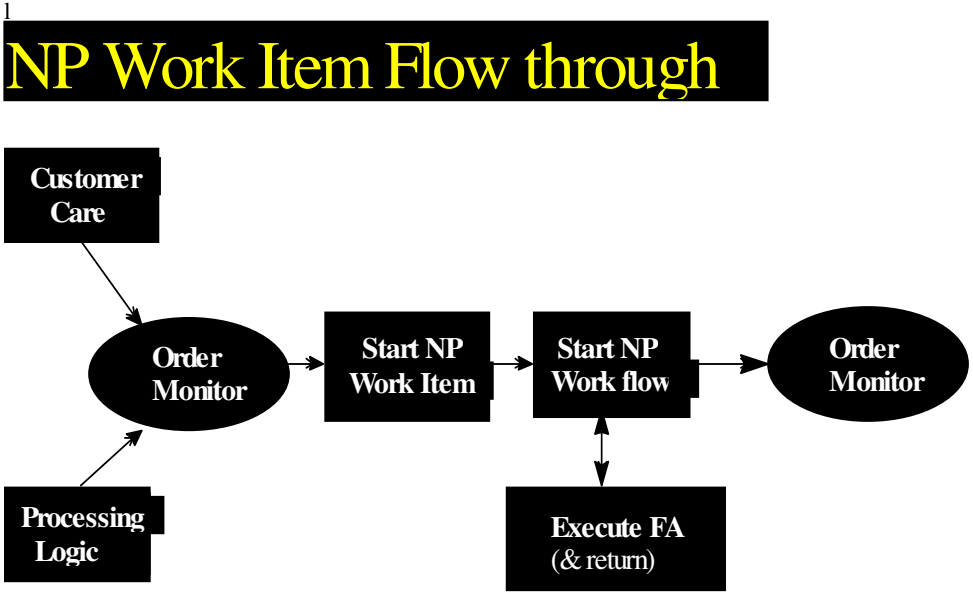
Work Item parameters act as a parameter pool for all the defined Work Items in SDP. New SDP Work Items can be defined using the existing parameters in the pool. New Work Item parameters can also be added to define new Work Items or extend existing Work Items. The table below gives the list of Work Items pre-configured by SDP for Number Portability. Every NP Workflow executes within the context of an SDP Order's *WorkItem* process. Each order has an unique id called the ORDER_ID and is supplied and maintained by the SDP engine. Similarly, each WorkItem too has an unique id called the WORKITEM_INSTANCE_ID which again is supplied by the SDP's order processing engine.

Orders can be submitted to the SDP-NP system either through the user interface or through a message. In the latter case, the message is processed and submitted as an order to the SDP-NP System as part of the message's processing logic.

The above figure shows the two possible ways in which an NP workflow could be kicked off i.e. through the Customer Care (i.e. UI) or the processing logic. Subsequently, the order tracked by the order monitor, which starts-up the NP workitem workflow giving it an unique id. As part of the business process execution, the workflow could execute a fulfillment action through the SDP's provisioning engine and waits for the response to get the control back. At the end of the workflow, the control is returned back to the order monitor.

See Appendix C: Work Item Parameters

Maintaining State In NP Applications



How is State Maintained in NP Applications?

All NP business processes involve messaging transactions with external systems. So, during a business process, the workflow could Send messages and also await for responses. The messages awaited could be either an immediate (positive or negative) ack to an earlier message sent by this workflow or could be some message which can be anticipated during the business process. In either case, every workflow needs to maintain state of the transaction while waiting for external messages. This is done by having a way to identify the workflow transaction with some unique identifier. This unique identifier is the *REFERENCE_ID*. To accommodate a similar identifier for the external system too, we have an *OPP_ REFERENCE_ID* (opposite party's Reference Id). Every incoming or outgoing NP message must contain these two (i.e. REFERENCE_ID and OPP_REFERENCE_ID) as mandatory elements in the header.

While awaiting a response from the external system, the workflow is uniquely identified by the combination of a) Message (code) awaited AND b) Reference Id of the transaction

Once this combination is matched, the appropriate workflow is called-back by the event manager.

How to decide on the Reference Id for an NP workflow?

The best choice for the Reference Id would be some unique identifier provided by a system which is external to NP. The reference id could be a 'Porting Id' if that uniquely identifies a porting transaction workflow and is guaranteed to be part of the responses received from the external system.

In the absence of any such identifiers from the external system, the NP business analyst must be able to arrive at such an identifier. Since the WORKITEM_ INSTANCE_ID is unique for that workflow, it could be a candidate for the Reference id. Another choice for reference id could be the ORDER_ID if no two workitems within the same order would register for the same message at the same time. Please note that the in the absence of any single value for the reference id, the workflow can use a concatenation of values e.g. it could be the concatenation of the starting number, ending number and porting id with some delimiters.

The workflow gets called back by the event manager when the response arrives if the *received message's REFERENCE_ID* field matches the *registered reference id*. To get the right workflow which maps to this (msg code + reference id) combination, the event manager searches through all the registrations. This searching by the event manager happens after the received message is *validated*. So, it needs to be ensured that at the end of the validation, the message header's REFERENCE_ID matches the registered reference id for the received message.

In cases, when the received message is not request's response but anticipated along the NP's business flow the external system would have no idea of our Reference Id. In such cases, the message validation logic could handle the populating of the header's REFERENCE_ID with the registered value. This way, the event manager would still be able to map the received message to the right workflow.

See Appendix B: Sample Workflows

Adapters in Oracle Number Portability

You configure an adapter to interact with the external system and to route messages to and from the external system appropriately. You configure an adapter by specifying its attributes.

The application comes with the file adapter already defined. This adapter enables messages to be passed to local files, or sent to a remote location using the provided FTP client.

Outbound Messages

For outbound messages, the adapter receives the message from the Oracle Advanced Queue, and generates the channel information accordingly.

According to the need, the adapter sends the message to one of the following channels:

- A local file
- A remote system (using File Transfer Protocol)

Inbound Messages

For inbound messages, the adapter acts like a server. It retrieves the message by monitoring a designated file directory. It then pushes the messages in the files it retrieves to the queue. MSG_SERVER is the consumer for these inbound messages.

Channels

A channel defines the path a messages takes during message transfer from, or to, a clear destination. You can only have one channel open per message at any given time. If you want to operate a new channel, then you must close the old channel. For example, if you open a file and send messages to it, then you must close that file before you can open another file channel.

File Adapter Attributes

The table following lists the attributes you must define for a file adapter.

File Adapter Attributes

Display Name	Internal Name	Description
Home Directory	HOMEDIR	The directory on the local machine where the file is to be created.
In Directory	IN_ARCHIVE_DIR	The directory to which the file is to be archived after the adapter reads it.
IP Address	IP_ADDRESS	IP address of the remote machine to which the file is to be sent.
Out Directory	OUT_ARCHIVE_DIR	The directory in which the file is to be archived after the file transfer has completed.
Password	PASSWORD	Password for authentication.
PORT	PORT	The port on which the remote FTP server is running. The standard port is 21.
Remote Directory	REMOTEDIR	The directory on the remote machine where the file is to be placed.
Scan Directory	SCANDIR	The directory from which files are read and sent to the application for processing.
User ID	USER_ID	The user ID for the authentication.

Fulfillment Element Types in Oracle Number Portability

Every fulfillment element belongs to a fulfillment element type group. The application supports the creation of new element types, if necessary.

Typically, you need fulfillment element types that correspond to the following categories:

- SCP (service control point)
- SSP (signal switching point)
- STP (signal transfer point) types
- SP (service providers) gateways
- NRC (number registration center) gateways

Predefined Fulfillment Element Types

The Oracle Number Portability application comes with a number of predefined fulfillment element types. The table following lists these predefined element types.

Predefined Element Types in Oracle Number Portability

Element Type	Description
BILLING_GATEWAY	Refers to billing system gateway
DIRECTORY_GATEWAY	Refers to Directory assistance gateway
NRC_GATEWAY	Refers to the Number Registration Center gateway
SCP	Refers to the LNP database for all service control points
SP_GATEWAY	Refers to messaging gateways for remote systems of other Service providers
SSP	Refers to signal switching point
STP	Refers to signal transfer point

Note: If the fulfillment element is a network element of type SCP, SSP, or STP, then you must define the number ranges served by this network element for a given feature type.

Fulfillment Elements in Oracle Number Portability

A fulfillment element is either one of the following:

- A network element
- An external system

For example, Service Switching Points and Signaling Transfer Points are types of network elements typically provisioned for number portability.

Fulfillment Elements as Network Elements

You configure each network element as a fulfillment element. This process includes specifying the following items for the fulfillment element.

- Its network ID
- Its Network Element Type
- Its attributes
- Its software versions
- The supported communication protocols
- The adapter type

Fulfillment Elements as External Systems

Every service provider (external system) with which the application interacts must have a corresponding fulfillment element created for it. This fulfillment element represents the configuration setup for the gateway to that external system.

- You **must** define the appropriate fulfillment element in order for the application to interact with an external system.
- You **must** configure adapters that can interact with that external system. Adapters route messages sent to and from the external system appropriately.

Understanding Fulfillment Actions

Fulfillment actions apply fulfillment procedures to fulfillment elements.

Each fulfillment element has a software version and an adapter assigned to it, and is of a certain fulfillment element type. From this information, the fulfillment action determines which fulfillment procedure to perform on this fulfillment element.

- One fulfillment action is written for every action that is performed on a fulfillment element.
- A fulfillment action acts upon only a single fulfillment element. (This is true at runtime only.)
- A fulfillment action can act on multiple fulfillment element types, however, by specifying for each element type the adapter, the software version, and the procedure to be applied.
- If fulfillment actions are defined for use in Oracle Number Portability, then they are assigned internal names such as Provision Number Portability, De-provision Number Portability, or Modify Number Portability. Display names correspond to these internal names. A user may define fulfillment actions, as well.

Predefined Fulfillment Actions

Oracle Number Portability comes with a number of fulfillment actions predefined. In most cases, it is not necessary to define new actions. These predefined fulfillment actions are listed in the following table.

Fulfillment Actions Predefined in Oracle Number Portability

Deprovision	Provision	Modify
DEPROVISION_CLASS	PROVISION_CLASS	MODIFY_CLASS
DEPROVISION_CNAM	PROVISION_CNAM	MODIFY_CNAM
DEPROVISION_ISVM	PROVISION_ISVM	MODIFY_ISVM
DEPROVISION_LIDB	PROVISION_LIDB	MODIFY_LIDB
DEPROVISION_NP	PROVISION_NP	MODIFY_NP
DEPROVISION_RN	PROVISION_RN	MODIFY_RN
DEPROVISION_WSMSC	PROVISION_WSMSC	MODIFY_WSMSC

Fulfillment Actions

Oracle Number Portability 11i also provides a predefined set of fulfillment actions which can be used to build a number portability application. Each Fulfillment action performs one of the three basic operations i.e., Provision, De-Provision or Modify the appropriate feature on a chosen fulfillment element. A Fulfillment action can be executed from Workflow. Since a fulfillment action maps to one or more Fulfillment Procedures, the developer is required to write the Fulfillment Procedures. Refer to the provisioning guide for more information on Fulfillment Actions and Fulfillment Procedures. The following section describes the Fulfillment Actions shipped as part of the NP 11i.

Fulfillment Action Name	Internal Name	Display Name
Provision NP	PROVISION_NP	Provision NP Fulfillment Elements. This will include the SSP, STP and SCP.
Deprovision NP	DEPROVISION_NP	Deprovision NP Fulfillment Elements
Modify NP	MODIFY_NP	Modify NP Fulfillment Elements
Provision CLASS	PROVISION_CLASS	Provision CLASS Fulfillment Elements
Deprovision CLASS	DEPROVISION_CLASS	Deprovision CLASS Fulfillment Elements
Modify CLASS	MODIFY_CLASS	Modify CLASS Fulfillment Elements
Provision CNAM	PROVISION_CNAM	Provision CNAM Fulfillment Elements
Deprovision CNAM	DEPROVISION_CNAM	Deprovision CNAM Fulfillment Elements
Modify CNAM	MODIFY_CNAM	Modify CNAM Fulfillment Elements
Provision LIDB	PROVISION_LIDB	Provision LIDB Fulfillment Elements
Deprovision LIDB	DEPROVISION_LIDB	Deprovision LIDB Fulfillment Elements
Modify LIDB	MODIFY_LIDB	Modify LIDB Fulfillment Elements

Provision ISVM	PROVISION_ISVM	Provision ISVM Fulfillment Elements
Deprovision ISVM	DEPROVISION_ISVM	Deprovision ISVM Fulfillment Elements
Modify ISVM	MODIFY_ISVM	Modify ISVM Fulfillment Elements
Provision WSMSC	PROVISION_WSMSC	Provision WSMSC Fulfillment Elements
Deprovision WSMSC	DEPROVISION_WSMSC	Deprovision WSMSC Fulfillment Elements
Modify WSMSC	MODIFY_WSMSC	Modify WSMSC Fulfillment Elements
Provision RN	PROVISION_RN	Provision RN Fulfillment Elements
Deprovision RN	DEPROVISION_RN	Deprovision RN Fulfillment Elements
Modify RN	MODIFY_RN	Modify RN Fulfillment Elements

Understanding Fulfillment Procedures

A fulfillment procedure is composed of specific commands that are sent to a fulfillment element when a fulfillment action is invoked. Examples of such commands might be to open a Telnet session or to provide element-specific routing commands to the element.

3. A fulfillment procedure comprises the following elements:

- A fulfillment element type
- An adapter
- The software version to be used on this fulfillment element type
- The fulfillment action to be performed

Creating a Fulfillment Element

The following provides an overview of the steps required to implement an SDP Fulfillment Action from within a User Defined Workflow.

1. Please read and follow the process outlined in *"Configuration HowTo, Service Delivery Platform, User Defined Workflow"* before continuing.
2. Using SDP, define a Fulfillment Element (FE) Type and Fulfillment Element (FE)
3. Define a Fulfillment Action (FA) and Fulfillment Procedure (FP)
4. Create a Fulfillment Element (FE) adapter
5. Using Oracle Workflow (WF) Builder, open the data store defined in step 1 above
6. Add an SDP Fulfillment Action (FA) to the WF process diagram

Process Details

The following provides a detailed description of the steps defined in section **Process Overview**

Using SDP

1. Logon to SDP
2. Access the Fulfillment Element screen from the "Service" tab

Steps to Define a Fulfillment Element

1. Click "Add"
   - Specify a Display Name
   - Specify a Internal Name
   - Define a Fulfillment Element Type
   - Click "Define Types" from the "Define Fulfillment Elements" screen
   - Click "Add"
   - Specify a Display Name
   - Specify an Internal Name
   - Specify a Description

2. Click “Attributes” tab
   - Specify a Display Name
   - Specify a Display Seq
   - Specify an Internal Name
   - Specify a Description
   - Specify a Default Value
   - Indicate if the Default Value is hidden or not

   Note: Repeat the above points for the number of Attributes required for the Fulfillment Element (FE) Type..

3. Click “Software” tab
   - Specify a Display Name
   - Specify an Internal Name
   - Select the required Adapter Type from the LOV
   - Specify a Description
   - Define a Connect procedure for the Fulfillment Element (FE) Type
   - Define a Disconnect procedure for the Fulfillment Element (FE) Type

4. Click “OK”

Note: The “Define Fulfillment Elements” screen will be displayed.

- Select a value from the Type LOV. The value selected here will be the Fulfillment Element (FE) Type defined in step 4 above.
- Specify a Description
- Specify a Min connection value
- Specify a Max connection value

Note: The Min and Max connection values indicate the number of Fulfillment Element Adapters that can be created for the Fulfillment Element (FE).

---

5. Click “SW Version”
   - Select a value from the Software Version (Adapter Type) LOV. The value selected here will be the Fulfillment Element (FE) Type software version defined in step 4 above.
   - Specify a Begin Date
   - Define a Connect procedure for the Fulfillment Element (FE). You can also select the Connect procedure defined for the Fulfillment Element (FE) Type in step 4 above.
   - Define a Disconnect procedure for the Fulfillment Element (FE). You can also select the Disconnect procedure defined for the Fulfillment Element (FE) Type in step 4 above.
6. Click “Attributes” tab
   - Specify the required attributes for the Fulfillment Element (FE).
   - Specify a Value for the attribute
7. Click “OK”
8. Access the Fulfillment Actions (FA) screen from the “Service” tab
9. Define a Fulfillment Action (FA)

10. Click “Add”
    - Specify a Display Name
    - Specify an Internal Name
    - Specify a Version
    - Specify a Description
    - Define a Fulfillment Element (FE) Routing procedure. This procedure should make reference to the Fulfillment Element (FE) that was defined in section **Using SDP**, step 3.
11. Click “Parameters”
    - Select the required parameters from the “Available Parameters” pick list
    - For each parameter selected, specify a default value if required
12. Click “Fulfillment Procedures”
    - Select a value from the Fulfillment Element (FE) Type LOV. The value selected here will be the Fulfillment Element (FE) Type defined in section **Using SDP**, step 4.
    - Select a value from the Software Version (Adapter Type) LOV. The value selected here will be the software version for the Fulfillment Element (FE) Type defined in section **Using SDP**, step 4.
    - Define a Fulfillment Procedure (FP) for the Fulfillment Action (FA)
13. Click “OK”

Steps to Create a Fulfillment Element (FE) adapter

1. Access the Connection Management Utility screen from the "Administration" tab
   - Select the Fulfillment Element (FE) that was defined in section **Using SDP**, step 3
   - Select the "Adapter tab
   - Click "New"
   - Specify a Name
2. Click "OK"
3. After clicking "OK", SDP will start an adapter for the Fulfillment Element (FE). The above screen shows the adapter with a Current Status of "idle"
4. Close the SDP screen

Using Oracle Workflow Builder

1. Start the Oracle Workflow Builder
2. Open the data store that requires an SDP Fulfillment Action (FA).
3. Ensure that the following SDP item types are including within the data store opened in step 2 above:
   - SDP Lookup Codes
   - SDP Standard
4. Open the WF process diagram that requires an SDP Fulfillment Action (FA).

   Note: In the sample screens that follow, reference will be made to a dummy process called "Test FA Process".

5. Drag the "Execute Fulfillment Action" function from the SDP Standard Item Type into the Process Diagram
6. Create the transitions between the Execute Fulfillment Action and other activities in the WF process diagram

   Note: The Execute Fulfillment Action has the following three result types which needed to be mapped:
   - FA Status: Aborted
   - FA Status: Error
   - FA Status: Success

Set the properties of the Execute Fulfillment Action activity

1. Click the “Node Attributes” tab
   - Specify the “FA Name” attribute value

     IMPORTANT: The “FA Name” attribute must be the same as the Internal Name defined for the Fulfillment Action (FA) in section **Using SDP**.

   - Specify the “FE Name” attribute value

     IMPORTANT: If The “FE Name” attribute must be the same as the Internal Name defined for the Fulfillment Element (FE) in section **Using SDP**.

2. Click “OK”

Geographic Areas Defined

Geographic areas are used in Oracle Number Portability for the following purposes:

- To identify the areas covered by a Number Registration Center. The application uses this information to determine to which Number Registration Center a message is to be sent.
- To identify the areas covered by a number range. The geographic area of a number range can be sent to the Number Registration Center whenever a port order is submitted.
- To identify the areas that correspond to a routing number.
- To identify the areas covered by a Service Provider.

Elements of Geographic Area Definition

You can create new geographic areas or modify an existing area to meet your business needs. Each geographic area contains the following items:

- Area type: Geographic area types are user-defined. You must define the area type before beginning to define the geographic area. The application comes with some types already pre-defined.
- Code: The code for a geographic area can be defined by a publicly available directory.
- Name: The name of the geographic area describes the physical area involved. For example, the name of a city or a region.

> **Note:** In defining a geographic area, you can specify that other geographic areas are its children. Thus, if you want to create a hierarchy of geographic areas, then you do this by building the hierarchy from the top downwards starting with the largest geographic areas.

Service Providers

In order to provide number portability from one service provider to another, each of the service providers must be set up in Oracle Number Portability. This information is used by the Service Order Administrator and the Service Management System to create and activate service orders as they are received.

The precise tasks that you perform for a service provider depend on the purpose of your installation of Oracle Number Portability. The following table lists the three usage types for the application and describes the kind of information that you need for each type.

Types of Oracle Number Portability Installations

Installation Purpose	Tasks
Individual Service Provider	You must set up the following: ▪ Full information for yourself ▪ Basic information for all other service providers with whom you expect to interact
Number Registration Center	You must set up the following: ▪ Full information for yourself ▪ Information for all service providers in your region

Subscription Versions

Subscription versions are used to maintain the status of porting requests across orders. There are two types of subscription versions. They are:

- **Order Subscription Versions:** Typically created by the recipient service provider, donor service provider, and central system during the Service Order Administration phase of the porting process.
- **Network Subscription Versions:** Typically created by the recipient service provider, donor service provider, and central system during the Service Management System phase of the porting process.

The key components of a Order Subscription Version differ slightly from the key components of a Network Subscription Version.

Order Subscription Versions versus Network Subscription Versions

Component	Order Subscription	Network Subscription
Phase Indicator	Yes	No
Telephone Number	Yes	Yes
Routing Number	Yes	Yes
Status	Yes	No
Porting ID	Yes	Yes
Customer	Yes	No
Change Cause Code	Yes	No
Recipient Service Provider	Yes	No
Recipient Service Provider Due Date	Yes	No
Donor Service Provider	Yes	No
Donor Service Provider Due Date	Yes	No
Mediator Service Provider (e.g., Central System)	Yes	Yes
Provisioning Map & Provisioning Status	No	Yes

Oracle Number Portability creates a subscription version for each individual telephone number.

For example, if a range of telephone numbers is ported such as 1234567 through 1234569, the application creates three individual subscription versions, one for each number ported.

Notes

In general, the following is true about the two types of subscription versions:

1. Participants typically do not have an Order Subscription Version for a given telephone number.
2. Donor and Recipient Service Providers typically have both an Order Subscription Version and a Network Subscription Version for a given telephone number.
3. Central Systems typically have only Order Subscription Versions.

Order Subscription Versions

An order subscription version is typically created by the recipient service provider, donor service provider, and central system during the Initiation phase of the porting process.

Recipient Service Provider

After a customer calls a new service provider to request service for his/her existing telephone number, the following events generally occur:

- The recipient service provider creates an order with a porting request for the customer's telephone number.
- This order is passed from the ordering system to the Service Delivery Platform.
- The Service Delivery Platform recognizes the porting request on a specific line item of the order and triggers Oracle Number Portability.
- Oracle Number Portability executes the customized business process for a port-in request and creates a porting order called an Order Subscription Version.
- The recipient operator also sends an outgoing message to notify the central system about the porting request.

The Order Subscription Version is used by the recipient service provider to maintain status of the porting request throughout its life cycle.

Central System

When the central system receives an incoming message from the recipient operator for a new porting request, the following events generally occur:

- The central system creates a porting order called an Order Subscription Version.
- The central system also sends an outgoing message to notify the donor service provider about the porting request.

The Order Subscription Version is used by the central system to maintain status of the porting request throughout its life cycle.

Donor Service Provider

When the donor service provider receives an incoming message from the central system for a port-out request for its existing telephone number(s), the following generally occurs:

- The donor service provider creates a porting order called an Order Subscription Version.

The Order Subscription Version is used by the donor service provider to maintain status of the porting request throughout its life cycle.

Porting ID

The Order Subscription Version for the recipient operator, donor operator and central system is identified by a single porting identifier, the Porting ID number.

- This Porting ID is typically assigned by the central system.
- After assigning a Porting ID, the central system notifies both the recipient operator and the donor operator of the porting ID through messaging.
- A porting ID can reference multiple Subscription Versions. The application does not require a unique porting ID for Subscription Versions.

Network Subscription Versions

A network subscription version is typically created by the recipient service provider, donor service provider and participant service providers during the Activation phase of the porting process.

Recipient Service Provider

After a porting request is received, the following events generally occur:

- Prior to the due date of a porting request, the recipient operator must usually make changes to its network to activate service for the new customer.
- At this time, the recipient operator creates a Network Subscription Version using the same Porting ID as the one used to create the Order Subscription Version.

The Network Subscription Version is now used to maintain details and status of each network element that has been updated and the number portability data used to update each network element.

Donor Service Provider

After a porting request is received, the following events generally occur:

- Prior to the due date of a porting request, the donor operator must also make changes to its network to de-activate service for the new customer and transfer all incoming calls to the recipient service providers network.
- At this time, the donor operator creates a Network Subscription Version using the same Porting ID as the one used to create the Order Subscription Version.

The Network Subscription Version is now used to maintain details and status of each network element that has been updated and the number portability data used to update each network element.

Participant Service Providers

After a porting request is received, the following events generally occur:

- On the due date or within a specified time period after the porting due date, the participant service providers epochally receives a broadcast from the central system.

  The purpose of the broadcast is to notify all participant service providers that are affected by the customers port to make the necessary changes to their network elements so that all incoming calls to the customer are now delivered to the new service provider instead of the old service provider.

  For example, this broadcast message can contain the Porting ID of the Order Subscription Version created by the central system during the Service Management System phase of the porting process.

- At this time, the participant service providers create a Network Subscription Version using the Porting ID received in the broadcast.

The Network Subscription Version is now used to maintain details and status of each network element that has been updated and the number portability data used to update each network element.

Porting ID

The Network Subscription Version for the recipient operator, donor operator and participant service providers is uniquely identified by a single porting identifier.

This Porting ID is typically the same Porting ID that was assigned by the central system to uniquely identify the corresponding Order Subscription Version.

Defining a Service Provider

To define a service provider, perform the following steps.

Prerequisites

None

Steps

1. In the Navigator, choose **Setup > Service Definitions > Service Providers**.
2. Click the New icon on the toolbar to create a new service provider.
3. Enter a name for the new service provider.
4. Use the Details tab to enter basic information about the service provider.
5. Use the Contacts tab to enter contact information for the service provider.

   Note that it is possible to enter more than one contact for a service provider.
6. Use the Numbers tab to enter a number range for each service provider.

   Note that it is possible for one service provider to own a particular number range, but they may have chosen to assign it to another service provider. For the purposes of number portability, Oracle Number Portability always uses the assigned service provider.
7. Use the Filters tab to enter number filters for the service provider.

   A filter is the list of telephone number ranges in which a service provider is interested. You enter this information to create filters so that a Number Registration Center can broadcast your subscription version information selectively.
8. Use the Coverage tab to enter geographic areas supported by a Number Registration Center service provider.

   If you are setting up a Number Registration Center as one of the service providers on your system, then you must provide information about those geographic areas supported by this Number Registration Center. You do not have to perform this step for other types of service provider.

   - If you do not see the desired area name in the Available box, enter the name in the Filter field and click **Find** to search for it.
   - If the area is not defined yet, click **New Geographic Area** to open the Geographic Area Setup window and then add it. See Defining a New Geographic Area for details, if necessary.

9. Use the Routing# tab to enter routing numbers for the service provider.

   Each service provider designates certain number ranges for the purpose of routing ported calls.

   These routing numbers provide the necessary information for routing a ported call. The numbers are mapped to the service provider's network elements and to the Oracle Number Portability subscriber's dialing number.

   Routing numbers are provided by the Number Registration Center for service providers other than you or for those service providers on whose behalf you act. All other service provider's entries are added to and deleted from this table upon instruction from the Number Registration Center.

10. Click the Adapter Configuration tab to enter information about a service provider adapters.

    - Click **New Adapter Configurations** to create a new fulfillment element or to modify an existing fulfillment element, if necessary.

11. Close the window.

    You are prompted to save your changes.

Defining a Porting Status Type

A service order may have one of a number of different statuses throughout its life cycle. You can define statuses that reflect the terminology and business processes of your organization.

Prerequisites

Each porting status must be associated with one of the four phases or statuses.

- Inquiry
- Ordering
- Active
- Old

Steps

1. In the Navigator, choose **Setup > Service Definitions > Porting Status Types**.
2. Click the New icon on the toolbar to create a new porting status type.
3. Enter a value at the left in the Porting Status Code field.
4. Enter a display name and brief description for the new status.
5. Select a phase for the new status.
6. Enter the desired display sequence.
7. Check the Active box to enable the use of this status by Oracle Number Portability.
8. Close the window.

   You are prompted to save your changes.

Defining Number Ranges and the Network Map

You use the Network Map form window to perform the following two tasks:

- Defining the served number ranges for a given network element
- Defining a network map

Defining the Served Number Ranges

Perform the following steps to define the served number ranges for a given network element.

Prerequisites

You must define a fulfillment element before you can associate a range of telephone numbers to it.

Steps

1. In the Navigator, choose **Setup > Interface Definition > Fulfillment Elements**.
2. Select a fulfillment element from the list at the left.
3. Click **Network Map**.

   The Network Map for <fulfillment element> window opens (where <fulfillment element> is the name that you chose earlier.
4. Select the Served Number Ranges tab.
5. Enter a start and an end number to define the range of telephone numbers with which this fulfillment element is associated.
6. Chose a feature type from the list of values that matches your business process.
7. Select the check box for Primary if the defined number range is to be stored in the primary source for routing information.
8. Select the check box for Active, if the defined number range is to be stored in the active database (as opposed to a backup system).
9. Close the Network Map window.

   You are prompted to save your changes.

Defining a Network Map

Prerequisites

None

Steps

1. In the Navigator, choose **Setup > Interface Definition > Fulfillment Elements**.
2. Select a fulfillment element from the list at the left.
3. Click **Network Map**.

   The Network Map for <fulfillment element> window opens (where <fulfillment element> is the name that you chose earlier.
4. Select the Associated Network Elements tab.

   The information in this tab is for future use and is currently read-only.
5. Close the Network Map window.

Messaging in Oracle Number Portability

Oracle Number Portability uses messages to communicate with external systems and initiate new orders in Service Delivery Platform. For example:

- Oracle Number Portability sends an outgoing message requesting a number port with the Central System.
- The Oracle Number Portability workflow then waits for an inbound message that confirms the request to port.

This use of messaging integrates the use of workflows with external conditional elements.

Messages can trigger a process, or set of processes in an application. Applications send and receive messages asynchronously using the Event Manager.

Messages Defined

Messages are used to communicate between applications and systems. Service Delivery Platform messages are defined in the industry-standard XML format. When a message is compiled, the following functions are created:

- Send()
- Publish()
- Validate()
- Process()
- Default_Process()
- Create_Msg()

These functions are used to communicate with the Number Registration Center and other service providers as well as with any other external system or internal Operational Support Systems.

Also, messages can trigger a set of processes internally within the system.

Message Definition

You define messages in one of two ways:

- You can select from a set of predefined and available messages that are preceded in Oracle Number Portability.
- You can build and define any required messages through the iMessage Studio.

Message Elements

A message comprises one or more message elements. A message element consists of a set of attributes:

- Parameters
- Name
- Data Type
- Length
- Default
- Mandatory
- Sequence

For example, area may be a message element composed of length and width. The message element name is used as a tag within the XML message. Each message element has a data source which must be defined.

Messages versus Events

The table following lists the differences between messages and events.

Messages versus Events

Type	Description
Message	A request to, or a reply from, a single pre-defined destination
Event	Broadcast notification to zero or more destinations that need not be specified until runtime

How the Event Manager Handles Messages

The Event Manager handles all messages entering the application system. Incoming messages are one of the following types:

- Request messages
- Responses to request messages
- Event notifications from remote systems

Remote applications send request messages and register for response messages with the Event Manager. The remote applications use an Oracle Number Portability API to register for messages.

When a message arrives, the Event Manager delivers the message to all registered applications after executing the validation and processing logic defined for the message. If no application has registered for a message, the application's default processing logic for that message is executed after message validation.

Message Processing by the Event Manager

The Event Manger performs the following actions upon arrival of a new message in the system.

1. It first retrieves the message from the Inbound Message queue or the Internal Events queue.
2. It executes the validation logic for the message.
3. It then searches the callback registration table to determine all workflow instances registered for the received message code and reference ID combination.
   - If a match is found, the Event Manager executes the incoming processing logic for the message.
   - If no matches are found, the Event Manager executes the default processing logic for the message. It then retrieves the next message from the Inbound Message queue and starts the process over.
4. Finally, it attempts to deliver the message to the registered workflow instances. If the message is not successfully delivered, the message status is marked as failed for the registrant. This status can be seen in the Callback Registration data.

Configuring iMessage Subscription

Within Oracle Number Portability, you can set applications to be default subscribers to messages. This is configured in the iMessage Subscribers utility. You can also associate default subscribers with events. For example, when a message occurs, the Event Manager ensures that an outbound message is automatically sent to the subscriber identified by a fulfillment element.

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 the application before they can be linked as responses to a message.

Manually Driven Messaging

There are also a variety of APIs available in Service Delivery Platform to facilitate custom subscription, de-subscription, enqueueing and dequeueing. Consult the *Oracle Developing XML-based Message Based Applications* reference guide for a detailed definition of the available API calls.

The iMessage Studio

Use the iMessage Studio as a tool for developing message-based Service Delivery Platform applications. It provides the following functionality:

- The ability to develop a message-based application, while generating the code to construct, publish, validate, and process application messages.
- The ability to compile and test these messages.
- The ability to share messages between applications, allowing for their re-use in various applications.
- The means to prevent redefining the same message in various applications across the enterprise.
- The ability to generate procedures through its APIs and its run-time messages.
- The means to customize pre-processing of outbound messages and the post-processing of incoming messages from the Event Manager to external subscribers.
- The ability to define a message set for a particular event during processing of an order within the application.
- The ability to construct timers (delayed messages) for use within the application.
- The ability to define messages in the XML industry-standard format.
- The ability to create a series of message processes to communicate with external systems, Service Providers, and internal Operational Support Systems.

Messages, Events and Timers

You use the iMessage Studio to create the following message-based items:

- Messages
- Events
- Timers

The following table lists the three types and describes each one.

Messages, Events and Timers

Type	Description
Messages	A sequence of text characters that are used for communication between application systems. Messages fall into two categories: - Messages 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 running to relay the published event to the remote system. External applications can register for an event using the default subscribers screen. Oracle Number Portability explicitly supports only the XML format.
Events	Messages that are sent to external systems and that are received from external systems. Events are published to both external and internal applications.
Timers	Messages that have a time delay and a duration interval associated with them.

Timers Defined

Timers in Service Delivery Platform are used to handle events or processes that must occur at specified time intervals within the application.

In general, timers are used in either one of two ways:

- To perform a task once, after a delay
- To perform a task repeatedly, after a delay

> **Note:** Service Delivery Platform timers use the Oracle Advanced Queue to perform queue operations. See the Oracle Advanced Queue documentation set for more information.

Timers provide the ability to handle events or processes that have to occur at specified time intervals in business applications. Timers consist of a delay and interval. The delay represents the amount of time to wait before starting a timer. The interval represents the wait time for a timer.

Timers maybe associated with a whole workflow process (ie. Process Timer), a particular activity within a workflow (ie Activity Timer) or a message (ie. Message Timer).

Activity Timers

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 in which a Workflow activity must be completed within.

For example,

A Workflow activity could be defined to check the network capacity for a requested data service. The time period to perform the Workflow activity could be 30 minutes. An Activity Timer can be associated with the Workflow activity to handle the required business logic if the 30 minute timeframe for the activity is exceeded.

Process Timers

An 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 in which a Workflow process must be completed within.

For example,

A Workflow process could be defined to provide a requested data service. The time period to perform the Workflow process could be 1 day. An Process Timer can be associated with the Workflow process to handle the required business logic if the 1 day timeframe for the process is exceeded.

Message Timers

An 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 in which a message acknowledgment is received.

For example,

A Workflow process could be defined to setup an account for a network service. A message is sent to the associated network service administrator. If a reply is not received 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.

How Timers Work

After a timer is defined, and linked to an activity (an event), or a message, one of two things can then happen:

- The timer never expires
- The timer expires

The Timer Never Expires

The following sequence of events occurs if the activity to which a timer is linked completes normally, as scheduled.

1. You use the iMessage Studio to define a timer and associate it with an event, or a message.
2. The Timer Manager enqueues the message in the timer queue (the Oracle Advanced Queue).
3. At run time, the Event Manager treats the timer just like any other message, except that timers are enqueued with a delay and interval which specifies the interval of time after which the message is available for dequeueing.
4. The activity (event or message) to which the timer is linked completes normally.
5. The workflow application deletes any timers related to the response message.
6. Workflow processing continues.

The Timer Expires

The following sequence of events occurs if the activity to which a timer is linked does not completes normally, and the timer expires.

1. You use the iMessage Studio to define a timer and associate it with an event, or a message.
2. The Timer Manager enqueues the message in the timer queue (the Oracle Advanced Queue).
3. At run time, the Event Manager treats the timer just like any other message, except that timers are enqueued with a delay and interval which specifies the interval of time after which the message is available for dequeueing.
4. The timer interval expires.
5. The Timer Manager dequeues the message from the timer queue and enqueues it on the message queue.
6. Subscribers to this message periodically check the message queue for its presence.
7. Workflow processing continues.

Process Overview

The following provides an overview of the steps required to implement timers within SDP.

1. Using SDP, define an Activity, Process, Message timer and Message Acknowledgment in iMessage Studio
2. Generate the required lookups representing the timers defined in step 1 above
3. Using Oracle Workflow Builder (WFB), create a new data store based on the SDP workflow template
4. Define a new WF item type
5. Defined required attributes for the new WF item type
6. Define required functions for the new WF item type
7. Define required messages for the new WF item type
8. Define required notifications for the new item type
9. Define a WF process diagram
10. Save the WF into the database
11. Create the required PL/SQL Packages and Procedures
12. Using SDP, define a Work Item which maps to the WF
13. Define a service which maps to the Work Item defined in step 12
14. List Timer APIs

Process

The following provides a detailed description of the steps defined in section Process Overview

Using iMessage Studio

1. Logon to SDP
2. Access the iMessage Studio screen from the "Configuration" tab
3. Define an Activity Timer
   - Specify an Internal Name
   - Select "Timer" from the Type LOV
   - Specify a Display Name
   - Specify a Description
   - Select a value from the Priority LOV
   - Select "Timer Queue" from the Queue Name LOV
   - Specify a Responsibility
4. Click the "Data Source" tab
5. Define the Delay and Interval for the Activity Timer
   - Select the tree root for the Activity Timer (ie. In the figure above, ACTIVITY_TIMER_1 is selected).
   - Select "SQL Query" from the Data Source Type LOV
   - Select "One and Only One" from the Data Source Cardinality LOV
   - Specify the Source for the Activity Timer

     IMPORTANT: The source SQL structure for the Activity Timer needs to be of the following format:

   - SELECT D delay, I interval FROM dual

     Where:

     `D` is the delay integer value (seconds)

     `I` is the interval integer value (seconds)
   - Select the DELAY branch from tree root for the Activity Timer (ie. In the figure below, DELAY is selected).

- Specify a Reference for the DELAY branch

IMPORTANT: The Reference value DELAY branch needs to map to the select-list specified in step 4 above. The Reference value needs to be of the following format:

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

Where:

- `<acitivity_timer_name>` is the Internal Name defined in step 3 above
- `<select-list column name>` is 1st select-list value of the SQL source specified in step 4 above
- In the figure above, the Reference value for the DELAY branch is

  `xnp$activity_timer_1.delay`

- Select the INTERVAL branch from tree root for the Activity Timer (ie. In the figure below, INTERVAL is selected).
- Specify a Reference for the INTERVAL branch

IMPORTANT: The Reference value INTERVAL branch needs to map to the select-list specified in step 4 above. The Reference value needs to be of the following format:

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

Where:

- `<acitivity_timer_name>` is the Internal Name defined in step 3 above
- `<select-list column name>` is 2nd select-list value of the SQL source specified in step 4 above
- In the figure above, the Reference value for the INTERVAL branch is

  xnp$activity_timer_1.interval

6. Click the “Detail” tab
7. Compile the Activity Timer

8. Click the “Compile” button

   If all goes well you should be presented with the above figure.

9. Click the “View” button

10. Clicking the “View” button enables to you to examine the PL/SQL package that is generated as a result of compiling the Activity Timer.

11. Click the “Close” button

12. Click the “Apply” button

13. Define an Process Timer

    - Specify an Internal Name
    - Select “Timer” from the Type LOV
    - Specify a Display Name
    - Specify a Description
    - Select a value from the Priority LOV
    - Select “Timer Queue” from the Queue Name LOV
    - Specify a Responsibility

14. Click the “Data Source” tab

15. Define the Delay and Interval for the Process Timer

    - Select the tree root for the Process Timer (ie. In the figure above, PROCESS_TIMER_1 is selected).
    - Select “SQL Query” from the Data Source Type LOV
    - Select “One and Only One” from the Data Source Cardinality LOV
    - Specify the Source for the Process Timer

IMPORTANT: The source SQL structure for the Activity Timer needs to be of the following format>

- SELECT D delay, I interval FROM dual

  Where:

  `D` is the delay integer value (seconds)

  `I` is the interval integer value (seconds)

- Select the DELAY branch from tree root for the Process Timer (ie. In the figure below, DELAY is selected).
- Specify a Reference for the DELAY branch

IMPORTANT: The Reference value DELAY branch needs to map to the select-list. The Reference value needs to be of the following format:

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

Where:

- `<process_timer_name>` is the Internal Name defined in step 6 above
- `<select-list column name>` is 1st select-list value of the SQL source specified in step 7 above
- In the figure above, the Reference value for the DELAY branch is

  xnp$process_timer_1.delay

- Select the INTERVAL branch from tree root for the Process Timer (ie. In the figure below, INTERVAL is selected).
- Specify a Reference for the INTERVAL branch

IMPORTANT: The Reference value INTERVAL branch needs to map to the select-list specified in step 7 above. The Reference value needs to be of the following format:

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

Where:

- `<process_timer_name>` is the Internal Name defined in step 6 above
- `<select-list column name>` is 2nd select-list value of the SQL source specified in step 7 above
- In the figure above, the Reference value for the INTERVAL branch is

```
xnp$process_timer_1.interval
```

16. Click the “Detail” tab
17. Compile the Process Timer
18. Click the “Compile” button

If all goes well you should be presented with the above figure. If an error is encountered, please check that you have followed steps 1 - 4 above correctly.

19. Click the “OK” button
20. Click the “Apply” button
21. Define a Message Timer
    - Specify an Internal Name
    - Select “Message” from the Type LOV
    - Specify a Display Name
22. Specify a Description
    - Select a value from the Priority LOV
    - Select “Outbound Message Queue” from the Queue Name LOV
23. Click the “Apply” button
24. Define a Event for Message Acknowledgment and Timer
    - Specify an Internal Name
    - Select “Event” from the Type LOV
    - Specify a Display Name
    - Specify a Description
    - Select a value from the Priority LOV
    - Select “Internal Event Queue” from the Queue Name LOV

25. Click the “Apply” button
26. Define Event Subscriber which is associated with a Timer
27. Access the Event Subscriber screen from the “Configuration” tab

    Select the event defined in step 9 above
28. Click the “Timers” tab

    Select the Activity Timer defined in step 3 above from the Timer LOV
29. Click the “Apply” button
30. Define Event (Grouping of Message Acknowledgment and Timer)
31. Access the Event Subscriber screen from the “Configuration” tab

    Select the event defined in step 10 above
32. Click the “Responses” tab
    - Select “ACK” from the Event Code LOV
    - Select timer defined in step 3 above from the Event Code LOV
33. Click the “Apply” button

Generate The Required Lookups Representing The Timers

Invoke the load_sdp_lookups.sh script on the server. The syntax for invoking this script is:

```
load_sdp_lookups.sh <USERNAME> <PASSWORD> <WF USERNAME> <WF
PASSWORD> <DB CONNECT STRING>
```

Note: The SDP template can be found in the following directory

$SDP_TOP/xnp/bin

Using Workflow to Build Timers

1. Start the Oracle Workflow Builder
2. Open the SDP template `XNPWFSTD.wft`

> **Note:** The SDP template can be found in the following directory
>
> **$SDP_TOP/admin/import**

3. Select the following pre-defined SDP item types which are associated with the `XNPWFSTD.wft` template:
   - SDP Lookup Codes
   - SDP Standard
   - Standard
4. Save the WF data store to a new file. This prevents overwriting the SDP template `XNPWFSTD.wft`
5. Define a New Item Type
   - Specify an Internal Name
   - Specify a Display Name
   - Specify a Description
6. Press "OK"
7. Create 3 attributes for the Item Type you created in step 5 above

Use the following table as a guideline for attribute details

Internal Name	Display Name	Description	Type
ORDER_ID	SDP Order ID	SDP Order ID	NUMBER
LINE_ITEM_ID	SDP Line Item ID	SDP Line Item ID	NUMBER
WORKITEM_INSTACE_ID	SDP Work Item Instance ID	SDP work Item Instance ID	NUMBER

8. Press “OK”

IMPORTANT: DP requires that the above three (3) attributes be defined for each WF process. SDP will not function correctly without these attributes.

9. Define any other attributes that maybe required for the WF
10. Define any functions that maybe required for the WF.
    - Specify an Internal Name
    - Specify a Display Name
    - Specify a Description
    - Specify a Function. Note that the function must be of the following format

      <package_name>.<procedure_name>
11. Press “OK”
12. Define any messages that maybe required for the WF notifications
    - Specify an Internal Name
    - Specify a Display Name
    - Specify a Description

      If this message is to be used with a notification that requires a response then you need to specify details in the “Result” tab.
    - Specify a Display Name
    - Specify a Description
    - Specify a Lookup Type
    - Specify the Default Value type for the Message Result

13. Click “OK”

> **IMPORTANT:** If you make reference to any TOKEN substitutions within the message body, you need to define message attributes for each TOKEN specified in the message body.

14. Define any notifications that maybe required for the WF
    - Specify an Internal Name
    - Specify a Display Name
    - Specify a Description
    - Specify a result type if a response to the notification is required.

> **IMPORTANT:** If the message to be attached to this notification is expecting a response from the notification, then you need to specify a Result Type that is the SAME as the Lookup Type specified in the message “Result” tab. (see step 9 above).

    - Specify a message
15. Click “OK”
16. Define a process
    - Specify an Internal Name
    - Specify a Display Name
    - Specify a Description
17. Click the “Details” Tab

    Specify The Error Process as `XDP_ERROR_PROCESS`. The XDP_ERROR_PROCESS is optional. You can defined your own error process.

18. Construct the WF process diagram

- Double-Click the process defined in step 11 above
- Drag the “START” and “END” functions from the Standard Item Type into the diagram
- Drag the required functions and notifications defined in steps 8 and 10 above
- Drag the following functions from the SDP Standard Item:

Function Name	Description
Fire Timer	Function to start a Timer which can be a Process or Activity Level timer
Start Related Timers	Function to start a Timer which maybe an Activity Timer or a Timer associated with a Message
Send Message	Function to send a message which has a timer associated to it
Subscribe To Acknowledgments	Function to subscriber for possible results from the Send Message Function
Complete Work Item and Update Status	Manages the SDP order status

IMPORTANT: SDP requires that the “Complete Work Item and Update Status” function from the SDP Standard Item Type be included in all WF process diagrams that are to be executed by SDP. SDP will not function correctly if this function is omitted from the WF process diagram.

19. Create the transitions between each activity in the WF process diagram

> **IMPORTANT:** SDP requires that the "Complete Work Item and Update Status" function be the last activity prior to the "END" function in the WF process diagram. SDP will not function correctly unless the "Complete Work Item and Update Status" is defined this way. Please note the picture above. The functions are mapped to the "Complete Work Item and Update Status" function and the "Complete Work Item and Update Status" function is mapped to the "END" function.

20. Define the function properties for the Process Level Timer (ie. 1st Fire Timer function)

Set the Node Attributes for the 1st Fire Timer Activity to the following

Node Attribute	Value
Timer Name	Process Level Timer define in step 6 above
Callback Reference ID	CUSTOM
Custom Callback Reference ID	Workitem Instance ID

21. Define the function properties for the Activity Level Timer (ie. 2nd Fire Timer function)

Set the Node Attributes for the 2nd Fire Timer Activity to the following

Node Attribute	Value
Timer Name	Activity Level Timer define in step 3 above
Callback Reference ID	CUSTOM
Custom Callback Reference ID	Workitem Instance ID

22. Define the Send Message function properties

Set the Node Attributes for the Send Message function to the following

Node Attribute	Value
Event Type	Timer Message defined in step 9 above
Parameter List	
Receiver Name	Dummy - Not to be used
Adapter Name	Dummy - Not to be used
Callback Reference ID	CUSTOM
Custom Callback Reference ID	Workitem Instance ID

23. Define the Start Related Timers function properties

Set the Node Attributes for the Start Related Timers function to the following

Node Attribute	Value
Message Code	Timer Message defined in step 9 above
Callback Reference ID	CUSTOMER
Custom Callback Reference ID	Workitem Instance ID

24. Define the Subscribe to Acknowledgments function properties

Set the Node Attributes for the Subscribe to Acknowledgments function to the following

Node Attribute	Value
Event Type	Event Acknowledgment defined in step 10 above
Callback Reference ID	CUSTOMER
Custom Callback Reference ID	Workitem Instance ID

25. Save your work to a file
26. Save you work to the database using the WF schema user
27. Exit from the Oracle Workflow Builder

Develop PL/SQL for Workflow Functions

1. Open your favorite PL/SQL editor
2. Create the relevant packages and procedures for each function defined in section **Using Oracle Workflow Builder**, step 8
3. Use the following template as a guideline for developing your PL/SQL procedure for use by WF

```
<procedure_name> (itemtypeIN  VARCHAR2
                 ,itemkeyIN  VARCHAR2
      ,actidIN  NUMBER
      ,funcmodeIN  VARCHAR2
      ,resultoutOUT VARHCAR2)
     IS
       <local declarations…>
 IF (funcmode = 'RUN') THEN
                      <your RUN executable statements>
                      resultout := 'COMPLETE:<result>';
                      RETURN;
                  ELSIF …
                      …
       END IF;

    EXCEPTION
         WHEN OTHERS THEN
           WF_CORE.CONTEXT('<package_name>'
                          ,'<procedure_name>'
                          ,<itemtype>
                          ,<itemkey>
                          ,<TO_CHAR(actid)>
                          ,<funcmode>);
     END;
```

4. Compile the packages/procedures in the XDP schema
5. Grant execute privileges on each package/procedure to the Workflow user
6. Connect to the Workflow schema as the Workflow user

Create a synonym each package defined

Timer Elements

A timer consists of several mandatory elements, along with as many additional optional elements as necessary to perform the business process. The table below lists these elements and describes them.

Timer Elements

Element	Mandatory	Description
Message type	yes	Type is timer.
Timer name	yes	The display name for the timer.
Timer interval	yes	The period of time that the timer is active before it expires.
Timer delay	yes	The amount of time to wait before starting the timing period.(The default is zero.)
...	no	Other elements as needed. These elements can be related to a product type, customer category, or Service Level Agreement according to the business requirements of the users.

Note: All timer delays and intervals use seconds as the unit of measure.

Message Response Timers versus Window Timers

The Service Delivery Platform uses two types of timers. They are:

- Message response timers
- Window timers

Note: Service Delivery Platform timers use the Oracle Advanced Queue to perform queue operations. See the Oracle Advanced Queue documentation set for more information.

The following table describes each of the two types.

Service Delivery Platform Timer Types

Type	Description
Message response timers	These timers are used to respond to messages generated (typically) by workflow activity. The timer interval is used to determine the time allowed to receive a response for the message.
Window timers	These timers refer to a specific, well-defined period (or window) of time. For example, you can create a timer to signal the occurrence of some future date minus 24 hours (end date - 24). This is useful if you need to determine whether or not certain events have occurred prior to this time so that the application can continue with its business processes. In this case, the delay would be: delay = (end-date - 24) - current_time

Timers and Jeopardy Management

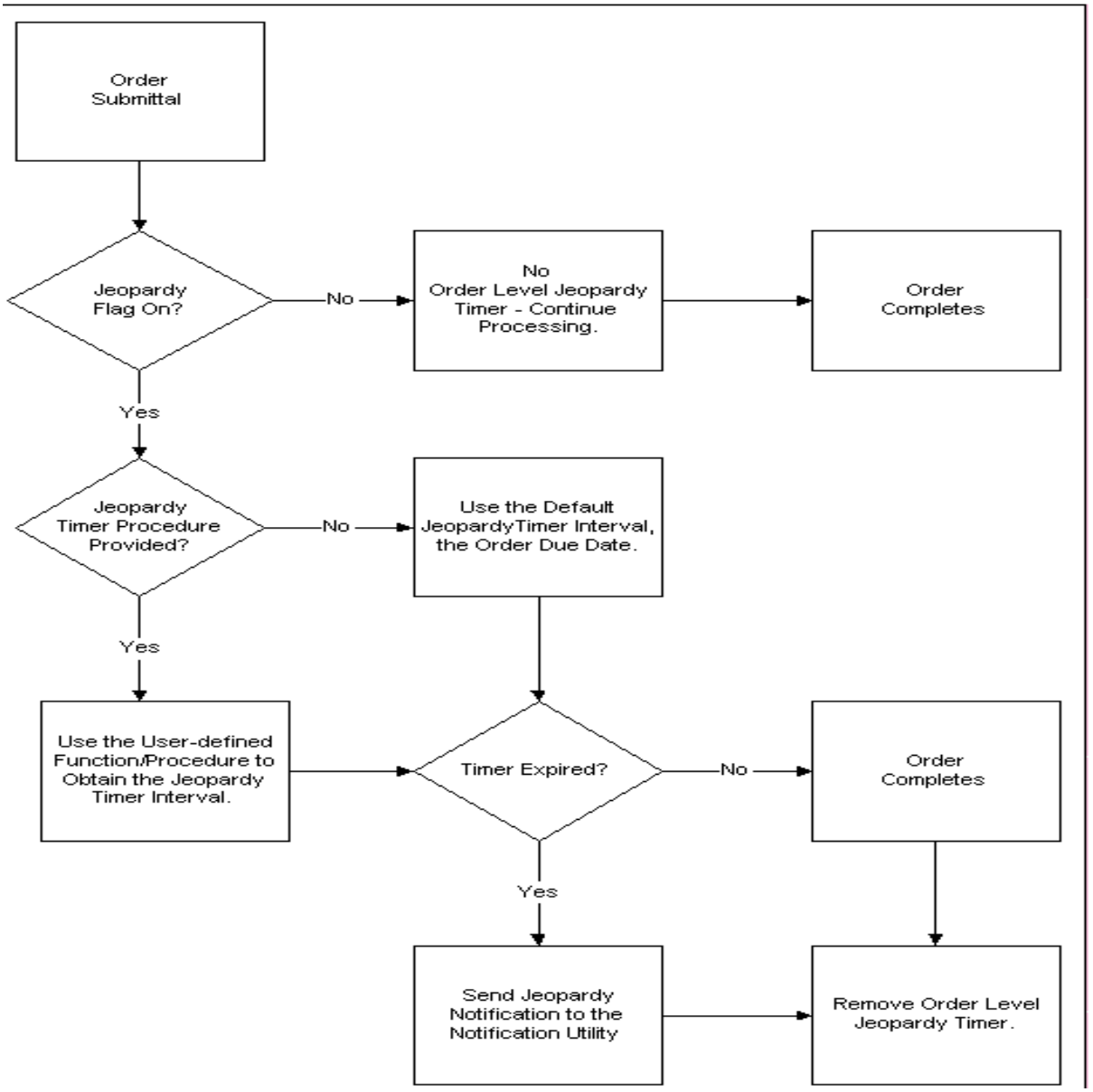
After a timer expires, the message to which it is linked becomes visible in the Oracle Advanced Queue, and the message is published to its subscribers.

This message can be used for a number of purposes, including the following:

- To notify the appropriate personnel to take any necessary action to resolve the jeopardy condition
- To initiate action within a workflow to manage the situation

These kinds of jeopardy management procedures are configurable by the user.

Jeopardy Notifications in Service Delivery Platform



Jeopardy Notifications

Jeopardy notifications provide messages to Service Delivery Platform users if either of the following occurs:

- A transaction becomes overdue
- A transaction may miss its assigned completion date

Order Level Jeopardy Notifications

Service Delivery Platform provides a seeded default timer that you can use to generate Order level jeopardy notifications. This timer is known as the Default Jeopardy Timer.

You may use either of the following methods to calculate the timer interval:

- The interval for the timer is calculated based on the Order Due Date (the internal date that stipulates the order completion date). This is the Service Delivery Platform default.
- Alternatively, you may retrieve the timer interval through a stored procedure. Service Delivery Platform provides a stubbed stored procedure for this purpose.

  If you provide the stored procedure, the stubbed procedure can be extended or used to call the user-defined stored procedure. In this case, use the JTF user hooks framework to enable this functionality.

For 11i, it should be possible to have an Order level jeopardy notification. A seeded default timer will be provided to enable this functionality. By default, the interval for the timer will be calculated based on the Order Due Date. This is the internal date to indicate the order completion date. The user can alternatively use a customized jeopardy timer.

```
If (Jeopardy Flag == 'Y')
 //This call to the seeded timer may be replaced by the customized timer
Seeded_Jeopardy_Timer.Fire(x_error_code);
WHEN Due_Date_Not_Found_Error;
RAISE_APPLICATION_ERROR(-#,<Textual description for Due Date Not Found>);
 //Associate exception with #
```

The Seeded Jeopardy Timer invokes a function to obtain the interval for the timer based on the Due Date for the order.

```
 If (Due_Date NOT FOUND)
 RAISE Due_Date_Not_Found_Error; //This exception trickles up
        Else
Interval = (Order Due Date - Sysdate)*24*60*60;
```

This code may be placed in the Post section of the Process Order API. The post section of the API is completely customizable by the user.

The default processing logic for the Seeded Jeopardy Timer will be used to invoke a Jeopardy Timer Notification workflow process that will notify the FMC. A workflow process will be created to handle this.

The user will have the option of firing the Jeopardy timer for each order by passing a 'Y' or 'y' value to the Jeopardy Enabled Flag in the XDP_OE_ORDER_HEADERS. The Insert_OE_Order API will check for the value of the Jeopardy Enabled flag. If the value of the flag is 'Y' or 'y', then the jeopardy timer will be triggered.

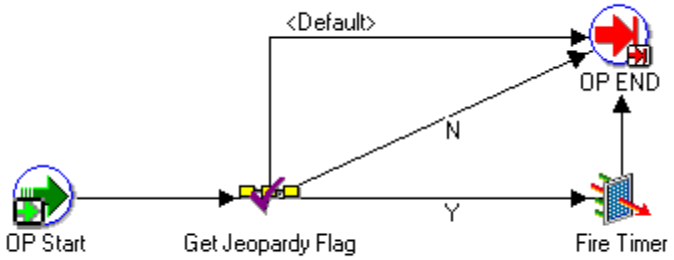
The jeopardy timer needs to be removed as soon as the order is completed. The Remove_Timer API will be called to accomplish this. The order_id and default jeopardy timer name will be passed to the Remove Timer API call.

The default processing logic for the jeopardy timer will send a notification to FMC to indicate that a jeopardy timer for a particular order id has expired. Workflow notification will be used to pass the default message to FMC. There are no jeopardy management processes associated with this timeout notification in SDP. However, users may configure customized jeopardy management processes based on the jeopardy timeout functionality.

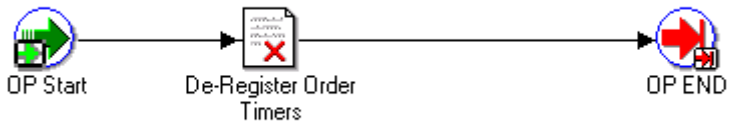
Forms Impact: XDPORD.fmb. Add a Jeopardy Flag indicator to the form.

Workflow: A default Jeopardy Timer Notification process will be created. This will be invoked from the default processing logic of the seeded Default_Jeopardy_ Timer. A new activity is created to check for the jeopardy flag related to the order id. If the jeopardy flag is 'Y', then the default jeopardy timer is invoked. The DeRegister activity is invoked in the post order section to deregister all timers for the given order id. The Customizable Oracle Provisioning Processes – Pre Order and Post Order are customized to incorporate this functionality.

Seeded Pre-Order SDP Customization



Seeded Post-Order SDP Customization



Configuring a Jeopardy Timer

The following steps describe the process of activating a jeopardy timer that is associated with a particular order:

1. You toggle the Jeopardy Enabled Flag in the XDP_OE_Order_Headers by passing it a **Y** or **y** value.
2. The Insert_OE_Order API checks for the value of the Jeopardy Enabled flag. If the value of the flag is **Y** or **y**, then the jeopardy timer is triggered.
3. Upon expiration of the jeopardy timer, the default processing logic for the timer sends a notification to the Notifications utility to indicate that a jeopardy timer for a particular order ID has expired. (Workflow notifications are used to pass the default message to Notifications module.)
4. You remove the jeopardy timer by calling the Remove_Timer API as soon as the order completes. In the call, you must pass the Order_ID and the name of the jeopardy timer to be removed.

The following diagram illustrates the process of configuring a jeopardy notification timer in Service Delivery Platform.

Workflow Customizations:

The user of the application is responsible for configuring this functionality.

Pre Order Processing - Check if Jeopardy Indicator is 'Y'(use the Get Jeopardy Flag activity). Fire default jeopardy timer if the indicator is 'Y'.

Post Order Processing - DeRegister timers using Order_id

The pre or post section of the Process Order public API within SDP may be used to start the jeopardy timer(seeded or user-defined).

Defining the Message Data Source

To set the source for the information in a message, perform the following steps.

> **Note:** Message elements that are not parameters require a data source.

Prerequisites

You must create the message details first, before you can define the message source. See Creating a New Message for details.

Defining Your Own Message Processing Logic

Incoming messages and events are handled by the Event Manager. There are multiple ways in which a message can be processed. To define the message processing logic, perform the following steps.

Prerequisites

You must create the message details first, before you can define the message processing logic. See Creating a New Message for details.

Guidelines

The following table lists the four types of message processing logic and provides a brief description of each.

Type	Description
Default Process Logic	If no application has registered for the message, the Event Manager automatically executes the default processing logic DEFAULT_PROCESS() for that message.
Validate Logic	The VALIDATE() procedure provides a hook to include business specific validation. It is automatically executed by the Event Manger on the newly arrived message.
Incoming Message Process Logic	The PROCESS() procedure also provides a hook to include the application logic. It is executed by the Event Manger before delivering the message to the callback procedure of the registered application.
Outgoing Message Process Logic	The outgoing process logic is executed before the message is put on the Outgoing Queue for delivery. The user-defined code is executed as part of the SEND() procedure.

Compiling iMessages

Perform the following steps to compile a message.

You can also select **Setup > Message Definition > Compile iMessage** to compile a single message, or **Setup > Message Definition > Compile All iMessages** to perform a batch compile.

> **Warning:** **Always ensure that no dequeuers or adapters are running when compiling messages in production.**

Prerequisites

The message must exist before you can compile it.

Sending a Test Message

You test how a message functions in Service Delivery Platform using the iMessage Studio **Test Message** interface. You use this utility to test sending a message to a queue. The message can also be tested using the standard SQL*PLUS interface.

You may use the Order Flowthrough utility, or the Workflow Monitor to monitor the progress of the message.

Errors during processing appear in the Notifications utility. Use this information to correct the error, then resubmit it to the queue for processing, if necessary.

> **Note:** The Test Message interface can be accessed either from the iMessage Studio or from the Test Center menu.

To test a message, perform the following steps.

Prerequisites

You must first compile the message before you can test it.

Steps

1. In the Navigator, choose **Setup > Message Definition > iMessage Studio**.
2. Select the Test Message tab.
3. Select the message code from the list at the left for the message you wish to test. The Parameter and Data Type fields fill in automatically.
4. Enter a value in the Reference Id field.
   This reference value relates the message being sent or received with the current application transaction. For example, this value could be the Order ID.
5. Enter a value in the Opposite Reference Id field.
   This reference value relates the message being sent or received with the remote application transaction. For example, this value could be the workflow instance.
6. Enter a (comma-separated) list of fulfillment element names in the Consumer List field to which the message or event will be sent (published).
7. Enter the Service Provider Code of the current service provider in the Sender Name field.

   For example, this value could be 9501.
8. Enter the Service Provider Code of the intended recipient service provider in the Recipient List field.

   For example, this value could be 9502. This field may be left empty for events that are being published.
9. Enter the version number of the message being sent.

   This value is incremented only if the original message cannot be processed due to some error.
10. Click **Send** to send the test message.
11. Click **OK** to exit.

    A confirmation window opens containing the Message ID and similar information. It is suggested that you manually record this data as it is useful in tracking a message within the system.

Working with Event and Timers

You perform a number of tasks relating to events and timers. These include the following:

- Creating a new event
- Creating a new timer
- Associating an event with a subscriber

Creating a New Event

To create a new event message, perform the following steps.

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 event message.
3. Select the Details tab.
4. Select Event from the drop-down list of available types.
5. Enter a short name for the event in the Display Name field.

   This name is referenced by the message studio when generating procedures.
6. Enter a brief description of the event.
7. Select a priority for the event.

   This value sets the priority for the event message in the outbound or inbound message queue.
8. Select a queue name from the drop-down list.

   Select either:

   - Inbound Message Queue
   - Outbound Message Queue

   If you are uncertain of which of these two to chose, then keep the default value unchanged.

9. Chose a user Responsibility for this message from the drop-down list.
10. Enter the DTD Location path.

    This value sets the path structure to the file that holds all the schema (Document Type Definitions) for this message. The file is named <message>.dtd, where <message> corresponds to the name of the message you are currently defining.
11. Close the window.

    You are prompted to save your changes.

Creating a New Timer

All timers must have a delay and an interval defined as elements. These two elements, Delay and Interval, can be one of the following:

- These elements can be default values.
- These values can be retrieved from a message store using a user-defined procedure.

Warning: Delay and Interval can not be parameters.

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.
3. Select the Details tab.
4. Select Timer from the drop-down list of available types.
5. Enter a short name for the event in the Display Name field.

   This name is referenced by the iMessage Studio when generating procedures.
6. Enter a brief description of the timer.
7. Select a priority for the timer.

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

8. Select Timer Message Queue from the drop-down list.
9. Chose a user responsibility for this message from the Responsibility drop-down list.
10. Enter the DTD Location path.

    This value sets the path structure to the file that holds all the schema (Document Type Definitions) for this message. The file is named <message>.dtd, where <message> corresponds to the name of the message you are currently defining.
11. Close the window.

    You are prompted to save your changes.

> **Note:** You can not add new elements to a timer.

Associating a Response with an Event

You can associate one or more responses with each defined event. These responses are used by the application workflow in the performance of activities.

> **Note:** If a message has an associated timer, the timer must be configured as a response for the event.

Prerequisites

You must first define an event before you can associate a response with it.

Steps

1. In the Navigator, choose **Setup > Message Definition > iMessage Subscribers**.
2. Select the Responses tab.
3. Select an event from the Events pane at the left-hand side of the window.
4. Chose an Event Code from the drop-down list.

    Note that it is possible to link multiple events to the selected event, if desired.
5. Close the window.

    You are prompted to save your changes.

Event for Timer or Message Acknowledgement

The Waiting For Acknowledgements activity subscribes to events. An event may be configured to include and Acknowledgement 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.

Registering Default Message Subscribers

To register message subscribers to receive automatic notification when an event occurs, perform the following steps.

Prerequisites

None

Steps

1. In the Navigator, choose **Setup > Message Definition > iMessage Subscribers**.
2. Select the Default Subscribers tab.
3. Select an event from the list at the left.
4. Chose a fulfillment element to associate with the event, using the drop-down list.
5. Add additional fulfillment elements, if desired, in the spaces provided.
6. Repeat steps 3 through 5, as many times as necessary.
7. Close the window.

   You are prompted to save your changes.

Note: You can also register message subscribers by registering an API using the Callback Registration window.

System Profile Options

The following are System Profile options for Oracle Number Portability

> **Note:** To set profile options, you must be logged in as System Administrator, **not** NP System Administrator.

The table following lists the profile options that you can set in the application.

Service Delivery Platform Profile Options

Profile Option Name	Description
ACK_REQD_FLAG	Ensures guaranteed delivery of messages. Adapters will send ACK back to calling program. The default is N.
DEFAULT_PORTING_STATUS	Initial porting status that is assigned to a porting record.
ENABLE_FEATURES	Determine if features such as CNAM, LIDB are available for Installation
ENABLE_NRC	Determines if this installation requires use of a central reference database.
INSTALL_MODE	Installation mode for operation of Number Portability
MAX_RETRIES	The maximum number of attempts that Send Message makes.
POP_TIMEOUT	Send Message Pop Time-out
SHOW_PROTECTED_MSG	Determines whether iMessage Studio Form displays seed messages
SP_NAME	Service Provider Code for Service Bureau mode.
TIMER_ACK_TIMEOUT_DURATION	Defines the Time-out duration while waiting for an Acknowledgment messages from the Remote System. Duration is in seconds.

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.

FNDLOAD Parameter List

Parameter	Description
apps/pwd	Specifies the APPS schema and password. ▪ 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. ▪ UPLOAD causes the specified data file to be uploaded to the database. ▪ DOWNLOAD causes the loader to fetch rows and write them to the specified data file.
configfile	Specifies the configuration file to use. ▪ 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. ▪ (DOWNLOAD) If the data file already exists, then it is overwritten. ▪ 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. ▪ 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). ▪ 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 XNP at the following location:

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

The table following lists the Loader files used with Oracle Number Portability 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	Parameters
xnpcbevt.lct	Callback Events	XNP_CALLBACK_EVENTS	MSG_CODE
xnpevtsb.lct	Event Subscribers	XNP_EVENT_SUBSCRIBERS	MSG_CODE
			FE_NAME
xnpgeoas.lct	Geographic Areas and Hierarchy	XNP_GEO_AREAS	CODE
		XNP_GEO_HIERARCHY	CODE
xnpmsgak.lct	Message Acknowledgments	XNP_MSG_ACKS	MST_CODE
xnpmsgps.lct	Messages	XNP_MSG_TYPES	MSG_CODE
xnpsnumr.lct	Served Number Ranges	XNP_SERVED_NUM_RANGES	STARTING_ NUMBER
			FE_NAME
			FEATURE_TYPE
xnpsptre.lct	Service Providers	XNP_SERVICE_PROVIDERS	CODE
xnpstats.lct	Porting Status Types	XNP_SV_STATUS_TYPES	STATUS_TYPE_CODE
xnptmrpb.lct	Timer Publishers	XNP_TIMER_PUBLISHERS	MSG_CODE

The following tables lists dependencies between the various files.

Loader File Dependencies

Name	Dependency
xnpcbevt.lct	First run xnpmsgps.lct
xnpevtsb.lct	First load the fulfillment element data
xnpgeoas.lct	First load the Areas, then the Hierarchy
xnpmsgak.lct	First run xnpmsgps.lct
xnpsnumr.lct	First load the fulfillment element data, then run xnpsptre.lct
xnptmrpb.lct	First run xnpmsgps.lct

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

Transferring Lookups to Workflow

To transfer common lookup codes from the configuration tables to workflow, perform the following steps.

Prerequisites

None

Steps

1. In the Navigator, choose **Setup > Application Definition > Transfer Lookups to Workflow**.

   The Load NP and OP Lookups onto Workflow window opens with the correct default values already set.

2. Click **Submit** to accept the defaults and start the transfer process.

3. Close this window when the process is complete.

Downloading Porting Lookups to File

To write the porting lookup codes from workflow into an ASCII file, perform the following steps.

Prerequisites

You must first transfer the lookups to workflow. See Transferring Lookups to Workflow for details.

Steps

1. In the Navigator, choose **Setup > Application Definition > Download Porting Lookups to File**.

   The Download NP Lookups from Workflow window opens with the correct default values already set.

2. Click **Submit** to accept the defaults and start the download process.

   This action retrieves the NP lookup codes from the database and stores them in file XNPDNPLK.wft.

3. Close this window when the process is complete.

> **Note:** To access the output files generated by the Lookup download, refer to the *Oracle Applications 11i System Administrator's Guide*, "Accessing Concurrent Manager Output and Log Files."

Downloading Common Lookups to File

To write the common lookup codes from workflow into an ASCII file, perform the following steps.

Prerequisites

You must first transfer the lookups to workflow. See Transferring Lookups to Workflow for details.

Steps

1. In the Navigator, choose **Setup > Application Definition > Download Common Lookups to File**.

   The Download OP Lookups from Workflow window opens with the correct default values already set.

2. Click **Submit** to accept the defaults and start the download process.

   This action retrieves the Service Delivery Platform lookup codes from the database and stores them in file XNPDOPLK.wft and XNPDNPLK.wft.

   The lookup codes in the generated files contain the Service Delivery Platform configuration information, including message definitions, fulfillment actions, and similar information. You can use this information for building your business processes.

3. Close this window when the process is complete.

Note: To access the output files generated by the Lookup download, refer to the *Oracle Applications 11i System Administrator's Guide*, "Accessing Concurrent Manager Output and Log Files."

Creating a Custom Notification Message

Perform the steps following to define a customized notification message that can be displayed at runtime.

Prerequisites

None

Steps

1. Create an ASCII text file as a container for the desired message.
2. Name the file with the following format:

   X%_NOTFN_%

   Do not exceed 29 characters total for the length of the file name.
3. Enter the text of your message.

   For example, this could be similar to the following message:

   Porting requested for &STARTING_NUMBER through &ENDING_NUMBER on &NEW_SP_DUE_DATE.

   The tokens after the ampersand (&) character are names of work item parameters. At runtime, the notifications utility scans the message and replaces the tokens with the values of the work item parameters.
4. Run the lookup loader script to load these user defined messages onto the workflow lookup code CUSTOMIZED_NOTN_MESSAGES.

The message utility sets the first line of the notification message to item attribute MSG_SUBJECT, and the entire contents of the file to item attribute MSG_BODY. The created notification contains the message with all the referenced work item parameters replaced with the actual value.

Message Processing Logic in Oracle Number Portability

Incoming messages and events are handled by the Event Manager in Oracle Number Portability. There are four possible ways that a message can be processed by the application.

They are:

- Default process logic
- Validate logic
- Incoming Message process logic
- Outgoing Message process logic

Following is a brief description of each type.

Note: If the user does not provide any message processing logic, the default is a NULL package body.

Default Process Logic

If no application has registered for the message, the Event Manager automatically executes the default processing logic DEFAULT_PROCESS() for that message.

The following example shows how to provide an application hook using the DEFAULT_PROCESS() procedure. Consider a case in which a PORTING_CONCUR message comes in asynchronously.

The default processing logic for this message is:

```
DECLARE
    l_telephone_num
VARCHAR2(10) ;
   l_clli
VARCHAR2(20) ;
    l_area_code
VARCHAR2(3) ;

BEGIN
     /* Reset error code and error message */
     x_error_code := 0 ;
     x_error_message := NULL ;
      /*
      Retrieve the Telephone number message element from the XML message
     */
     XNP_XML_UTILS.DECODE( p_msg_text,

'TELEPHONEí',
                                                                    l_
telephone_num)  ;
     /*
     Retrieve the central office or the CLLI on which it has to be provisioned
     */
     XNP_XML_UTILS.DECODE(  p_msg_text,
                                                                  'CLLI'
,
                                                                    l_
clli) ;
```

```
     /*
     Ensure that the right central office is used for provisioning
     */
     l_area_code := SUBSTR(l_telephone_num,1,3 ) ;

IF ((l_area_code = ë415í) AND
            (l_clli = ëSFOí)) THEN
     /* Customized procedure to provision the number */
     /* Not part of NP core functions */

 ELSE
     /*
     Customized procedure to notify the customer care system. Not part of NP
core functions
     */

     NOTIFY_CUSTOMER_CARE(l_tn,
                                                                          x_
error_code,
                                                                          x_
error_message) ;
     END IF ;
END ;
```

Validate Logic

The VALIDATE() procedure provides a hook to include business specific validation and is automatically executed by the Event Manager on newly arrived messages.

If no validation logic is specified, the procedure is created with a "NULL;" statement. The signature for this procedure is given in the following code.

> **Note:** The Event Manager will not process and deliver the message in case an error is returned in X_ERROR_CODE or X_ERROR_MESSAGE.
>
> However the resulting error code and error message is logged into the system log messages.

```
VALIDATE(
p_msg_header IN XNP_MESSAGE.MSG_HEADER_REC_TYPE,
p_msg_text IN VARCHAR2,
x_error_code OUT NUMBER,
x_error_message OUT VARCHAR2,
p_process_reference IN VARCHAR2 DEFAULT NULL ) ;
```

The following is an example of code that checks for a valid telephone number. (The use of XNP_XML_UTILS.DECODE works only in case of messages with no repeating elements.)

```
DECLARE
     l_telephone_num                                    VARCHAR2(10) ;
     l_service_provider                                   VARCHAR2(10) ;

BEGIN

     /* Reset error code and error message */

     x_error_code := 0 ;
     x_error_message := NULL ;

      /* Retrieve the telephone number */

     XNP_XML_UTILS.DECODE( p_msg_text,


'TELEPHONE' ,
```

```
                                                                    l_
telephone_num ) ;

     /* Retrieve the service provider */

     XNP_XML_UTILS.DECODE(  p_msg_text,

'SERVICE_PROVIDER' ,
                                                                    l_
service_provider )  ;
     /*
     Ccustom procedure to check if the telephone number is in the service
providerís defined number range
     */

     TN_RANGE.CHECK_SP_VALIDITY(  l_telephone_num,

l_service_provider,

x_error_code,

x_error_message ) ;

END ;
```

Incoming Message Process Logic

The PROCESS() procedure also provides a hook to include the application logic and is executed by the Event manager before delivering the message to the callback procedure of the registered application.

The following example code stores the PORTING_ID from an NPR_ACK for the recipient.

```
DECLARE
 l_REFERENCE_ID VARCHAR2(40) := NULL;
 l_porting_id VARCHAR2(40);
BEGIN

 x_ERROR_CODE := 0;

 /*
  * Get the OPP_REFERENCE_ID as in the received message
  * which is the workitem instance id.
  */
 XNP_XML_UTILS.DECODE
```

```
  (p_msg_text
  , 'OPP_REFERENCE_ID'
  , l_REFERENCE_ID
  ) ;

-- Get the NPR PORTING_ID
 XNP_XML_UTILS.DECODE
  (p_msg_text
  , 'PORTING_ID'
  , l_porting_id
  ) ;

  XDP_ENGINE.SET_WORKITEM_PARAM_VALUE
   (to_number(l_REFERENCE_ID)
   ,'PORTING_ID'
   ,l_porting_id
   ,NULL
   );

/* Set the reference to communicate with Number Registration Center, is the
PORTING_ID
*/
  XDP_ENGINE.SET_WORKITEM_PARAM_VALUE
   (to_number(l_REFERENCE_ID)
   ,'OPP_REFERENCE_ID'
   ,l_porting_id
   ,NULL
   );

END;
```

Outgoing Message Process Logic

The out process logic is executed before enqueueing the message for delivery No procedure is generated but the defined code is executed as part of the SEND() procedure Logging an out going message can be a good use of this hook

The following example code copies a message from the outbound to the inbound queue.

```
BEGIN
DECLARE
  my_header XNP_MESSAGE.MSG_HEADER_REC_TYPE ;
  my_xml  VARCHAR2(4000);
  BEGIN
  my_header := l_msg_header ;
```

```
  my_xml := l_msg_text ;
  XNP_MESSAGE.GET_SEQUENCE(my_header.message_id) ;
  my_header.direction_indr := 'I' ;
  XNP_MESSAGE.PUSH(p_msg_header=>my_header,
    p_body_text => my_xml,
    p_queue_name=>XNP_EVENT.C_INBOUND_MSG_Q,
    p_correlation_id=>'MSG_SERVER') ;
  END;
END;
```

Workflows

Workflows in Oracle Number Portability are sub-processes which execute during the order fulfillment process.

- Workflows are used to automate the business processes necessary to fulfill the order.
- Workflows are generally referred to as work items in the application.
- Workflows are created using the Oracle Workflow Builder.

Oracle Workflow Builder

You use the Oracle Workflow Builder to customize your business needs. Through workflow, you can route any type of information in an asynchronous manner, according to your business rules.

Within the Oracle Workflow Builder, you create, view, or modify a business process with simple drag and drop operations. In addition, you can create and modify all workflow objects, including activities, item types, processes and notifications.

At any time, you can perform the following operations on workflow objects:

- Add
- Remove
- Modify
- Set up new prerequisite relationships among the various types

You can easily work with a summary-level model of your workflow, expanding activities within the workflow as needed to greater levels of detail.

Reference

See Appendix B: Sample Workflows

See the Oracle Workflow documentation set for information relating to the use of workflow.

Workflow Activity Functions

When defining a workflow, it is important to understand the interaction between Oracle Workflow and Oracle Number Portability.

In simple terms, you build your workflow using any or all of the following types of function activities:

- Standard function activities provided by Oracle Workflow
- Standard function activities provided by Service Delivery Platform
- Standard function activities provided by Oracle Number Portability
- Your own user-defined function activities

Standard Function Activities Provided by Oracle Workflow

These types of function activities are actions that simply progress a workflow to the next activity, or change standard workflow information. For example, this could be merely the start or the end of a workflow.

- These functions are defined during workflow installation, in the Standards section.
- These procedures types are held in the WF_STANDARD package in any Oracle database.
- Do not adjust these procedures.

Standard Function Activities Provided by the Service Delivery Platform

Function activities in workflow are split into those activities specific only to the Service Delivery Platform, and those activities specific only to number porting.

- Function activities that are specific to the Service Delivery Platform are generally related to messaging, timers, synchronization, preparing notifications, work items, and fulfillment actions.
- Function activities that are specific to number porting are related to the number portability process.

This allows for a better visual understanding of the difference between the activities.

Certain actions that take place in a process require a check with, or an update to, the Service Delivery Platform configuration.

Several examples:

- If an activity in a workflow is to change the porting status of an order, a number of checks need to be made against the data held in Service Delivery Platform. In particular, the application must verify that the new porting status is a valid one.
- During Send Message a check must be made with the application's repository to ensure that the message is a valid message and that the parameters are correct.

The actual PL/SQL procedures for functions that are specific to Service Delivery Platform and Oracle Number Portability are held in the XNP_WF_STANDARD package.

The table following lists the Service Delivery Platform standard function activities supplied with the application.

Service Delivery Platform Standard Function Activities

Name	Description
Check Order Result	Determines the value of the of the ORDER_RESULT work item parameter, and completes the activity based on this value.
Complete Work Items and Update Status	Sets the work item status, and notifies the order tracking system (the Service Delivery platform), about the completion of the work item.
Execute Fulfillment Action	Executes the fulfillment action for the given feature type. It associates the fulfillment element to a work item, and submits it for provisioning. It then registers for an event to wake it up after the fulfillment action is complete.
Prepare Customized Notification	Defines a customized notification to be displayed at runtime.
Prepare Notification Message	Prepares the notification to be sent to the target.
Publish Event	Publishes a single business event. In the case of internal events, any recipients of this event must already have subscribed to the event.
Send Message	Sends a message to a single recipient. The Send procedure first checks whether or not an adapter is available for the recipient of the message. ▪ If an adapter is available, it proceeds with the send action. ▪ If an adapter is unavailable, a callback is registered to receive a notification once the adapter becomes available.
Set Order Result	Sets the value of the ORDER_RESULT work item parameter to the lookup value set by the workflow builder.
Subscribe to Acknowledgments	Identifies and registers for all expected responses messages. (In many cases, sending a request message can result in the generation of multiple response messages.)
Subscribe to Business Event	Registers a callback for the given event from the remote or local system.

Standard Function Activities Provided by Oracle Number Portability

The table following lists the Number Portability standard function activities supplied with the application.

Number Portability Standard Function Activities

Name	Description
Check if Donor Can Port Out	Checks whether or not the donor service provider of this porting transaction has provisioned the number range, or has assigned the number range. ▪ If either condition is true, then the activity completes with Y.
Check if Donor is Initial Donor Also	Checks whether or not this donor is also the initial donor.
Create Porting Order	Creates a Porting Record for each telephone number in the range. The Porting ID for the first record is the same as that set in the work item parameter.
Create SMS Porting Records	Creates a Porting Record for each telephone number in the range. The Porting ID for the first record is the same as that set in the work item parameter.
Deprovision FEs	Sets the fulfillment elements to be deprovisioned for this feature type and number range. For each fulfillment element, the Service Delivery Platforms's provisioning procedure (Execute FA) is invoked. At the end of this activity, control passes to the provisioning subsystem which executes the fulfillment procedure. An FA_DONE message is subscribed for each fulfillment action being executed which gives the execution result of the fulfillment procedure. The callback procedure associated with the FA_DONE handles the responses received. The immediate next activity following this activity must be the SDP Standard **Wait For Flow**. This is to ensure proper hand-off from the provisioning system back to the Number Portability system. Only fulfillment elements that were earlier provisioned by this service provider can be modified. Otherwise, the fulfillment elements are ignored.
Determine Current Service Provider Role	Determines whether or not a given service provider is the donor, original donor or recipient for the current porting transaction and completes the activity with the appropriate result code.
Determine If Subsequent Porting Request	Checks whether or not this is a subsequent porting request. ▪ Returns either Y or N.
Does Porting Record Exist for Donor	Check whether or not there exists a porting record with the given status in this telephone number range that belongs to the given donor's service provider ID. ▪ Returns either Y or N.

Number Portability Standard Function Activities

Name	Description
Does Porting Record Exist for Recipient	Check whether or not there exists a porting record with the given status in this telephone number range that belongs to the given recipient's service provider ID. Returns either Y or N.
Get Flag Value	Retrieves the Locked flag value for the given PORTING_ID work item parameter. ▫ The activity completes with a flag value of Y or N.
Get Porting Status	Retrieves the status of the porting record for the given Porting ID
Modify FEs	Retrieves the fulfillment elements to be modified for this feature type and number range. For each fulfillment element, the Service Delivery Platform's provisioning procedure (Execute FA) is invoked. At the end of this activity, control passes to the provisioning subsystem which executes the fulfillment procedure. An FA_DONE message is subscribed for each fulfillment action being executed which gives the execution result of the fulfillment procedure. The callback procedure associated with the FA_DONE handles the responses received. The immediate next activity following the activity must be SDP Standard **Wait For Flow**. This is to ensure proper hand-off from the provisioning system back to the Number Portability system. Only fulfillment elements that were earlier provisioned by this service provider can be modified. Otherwise, the fulfillment elements are ignored.
Provision FEs	Sets the fulfillment elements to be provisioned for this feature type and number range. For each fulfillment element, the Service Delivery Platform's provisioning procedure (Execute FA) is invoked. At the end of this activity the control passes to the provisioning subsystem which executes the fulfillment procedure. An FA_DONE message is subscribed for each fulfillment action being executed which gives the execution result of the fulfillment procedure. The callback procedure associated with the FA_DONE handles the responses received. The immediate next activity following this activity must be the SDP Standard **Wait For Flow**. This is to ensure proper hand-off from the provisioning system back to the Number Portability system.
Reject Message	Rejects a message.
Remove from SMS Provisioning Map	Deletes the fulfillment element mapping for this telephone number range from the application database.
Retry Message	Retries the message.

Number Portability Standard Function Activities

Name	Description
Set Flag Value	Sets the flag to the given value for the entities in XNP_SV_SOA for the given PORTING_ID work item parameter and FLAG_NAME for the current service provider. ▪ Sets a values of Y or N.
Update Charging Information	Updates the Subscription Version in the Service Order Administrator for each telephone number with the given Porting ID with the invoice information.
Update Comments and Notes Information	Updates the comments and notes for the current Porting ID and the current service provider.
Update Current SV Status	Updates the status type code in the XNP_SV_SOA with the new status type code. All records with the Porting ID and belonging to the current service provider are updated to the new status. If the new status belongs to the ACTIVE phase, and if there exists records for this number range already in ACTIVE phase, then these records are first reset to the OLD phase. The actual update of the records with the given porting ID is performed after this step.
Update Customer Information	Updates the customer information for the current Porting ID and the current service provider.
Update Network Information in SOA	Updates the network information in the XNP_SV_SOA for the current Porting ID and the current service provider.
Update Porting Status	Updates the status type code in the XNP_SV_SOA with the new status type code. All records with the porting ID and belonging to the current service provider are updated to the new status. If the new status belongs to the ACTIVE phase, and if there exists records for this number range already in ACTIVE phase, then these records are first reset to the OLD phase. The actual update of the records with the given porting ID is performed after this step.
Update SMS Provisioning Map	Updates the Provisioning Status of the fulfillment element for the given feature type and telephone number range.
Update with New Date	Updates the date for the porting record with the given Porting ID. The format used must be in the following format: ▪ YYYY/MM/DD HH24:MI:SS
Verify Porting Status	Checks if the STATUS_TYPE_CODE from XNP_SV_SOA for the given PORTING_ID is same as the given status type code (in STATUS_TO_COMPARE_WITH). ▪ Returns T if the two statuses match. ▪ Returns F if the two statuses do not match.

User-defined Function Activities

Some actions that take place in a process are user specific actions. These actions may, or may not, have standard workflow or Oracle Number Portability implications.

For example, it is possible that a number in a log table must be incremented every time a status change has occurred. As this activity is not an order specific action, this is merely an example of a procedure that retrieves the number from the table and adds one to it.

In contrast, a user-defined procedure contains some workflow or Oracle Number Portability API calls in it.

For example, returning to the previous log table example, it is possible that there is a requirement to log the From Status and the To Status when an order status change takes place.

In order to perform this action, a user-defined procedure must first retrieve the order parameters or the work item parameters using an Oracle Number Portability API call, and then insert them into the log table.

Another example of a user-defined procedure that is referenced by a workflow function is the process of retrieving the Existing Service Provider Name parameter from the Oracle Number Portability order and then using the retrieved value to determine the service provider identifier. The procedure code can then update the work item parameter SP_ID in the Oracle Number Portability order.

Considerations for Future Upgrade

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.

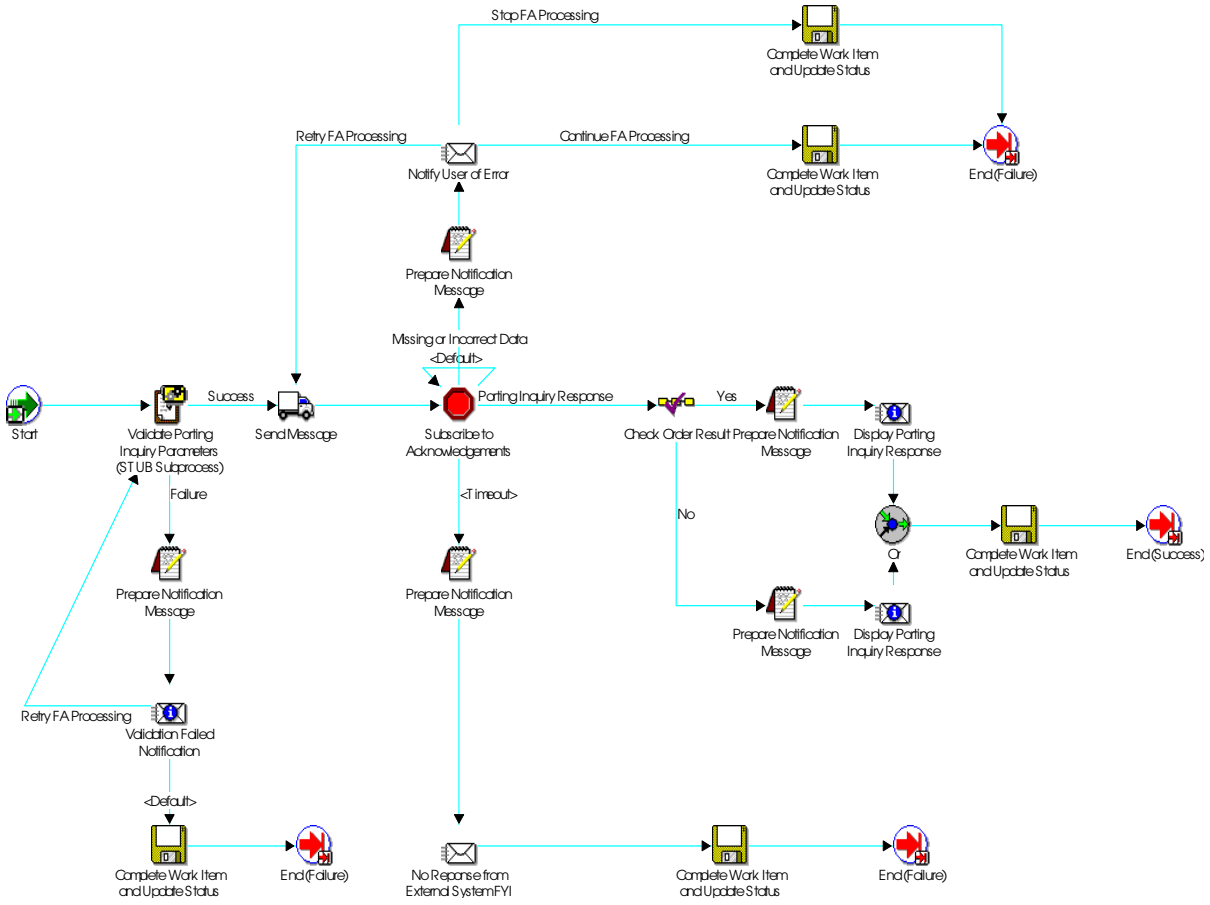
PROCEDURE Process_DRC_Order(

```
        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

Sample Workflows

B.0.1 Inquire Recipient Operator For Porting In



Internal Name:

PORTING_INQUIRY_FROM_OMS

Description:

New Operator receives porting inquiry from Order Management System. A porting inquiry is launched and a response is obtained from the Donor operator and the response is conveyed to the Order Management System.

B.0.2 Porting Order Initiated By Recipient Operator

Internal Name:

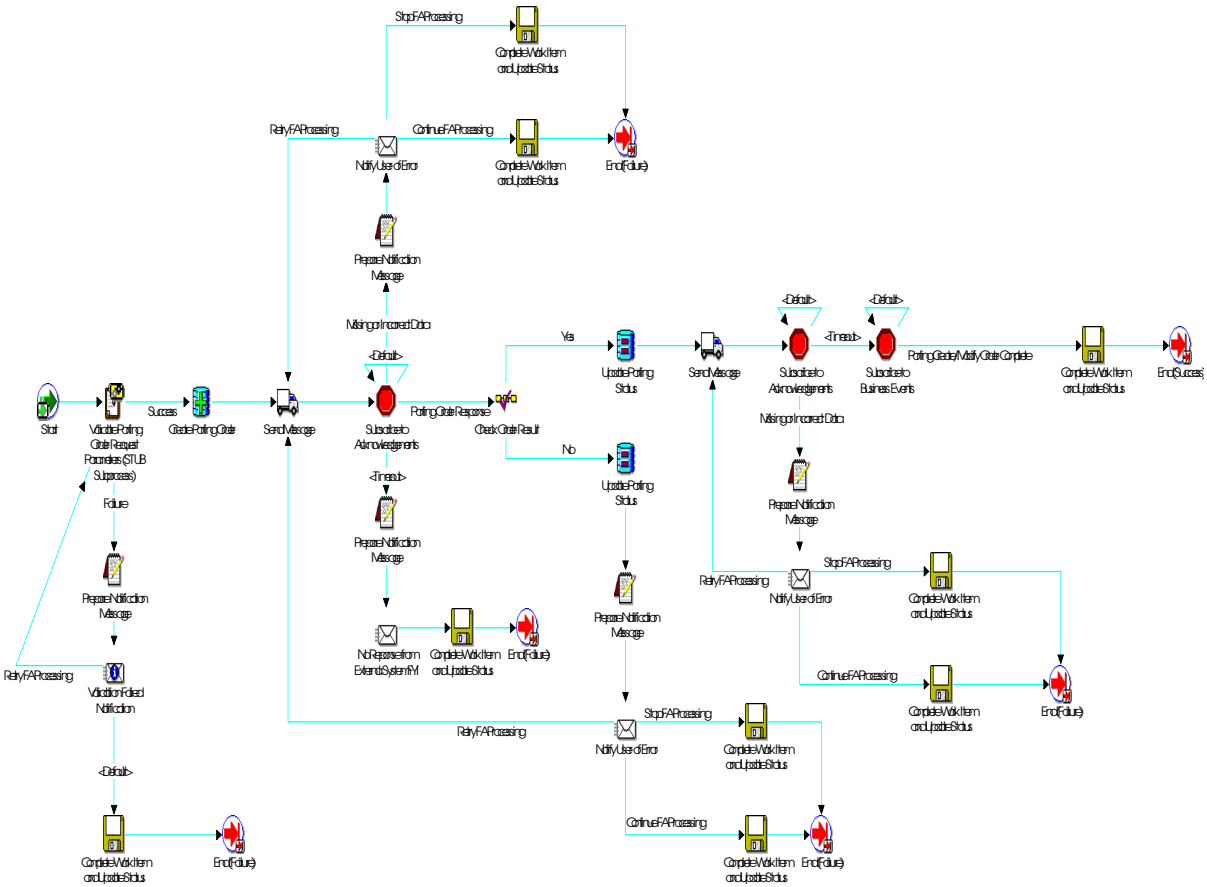
PORTING_ORDER_FROM_OMS

Description:

Recipient Operator receives porting order from Customer Care or an Order Entry System. A Porting Subscription Version is created and the Porting Order Inquiry is sent to the NRC. The process then waits for an approval or rejection from the NRC.

B.0.3 Inquire Recipient Operator For Porting In

:



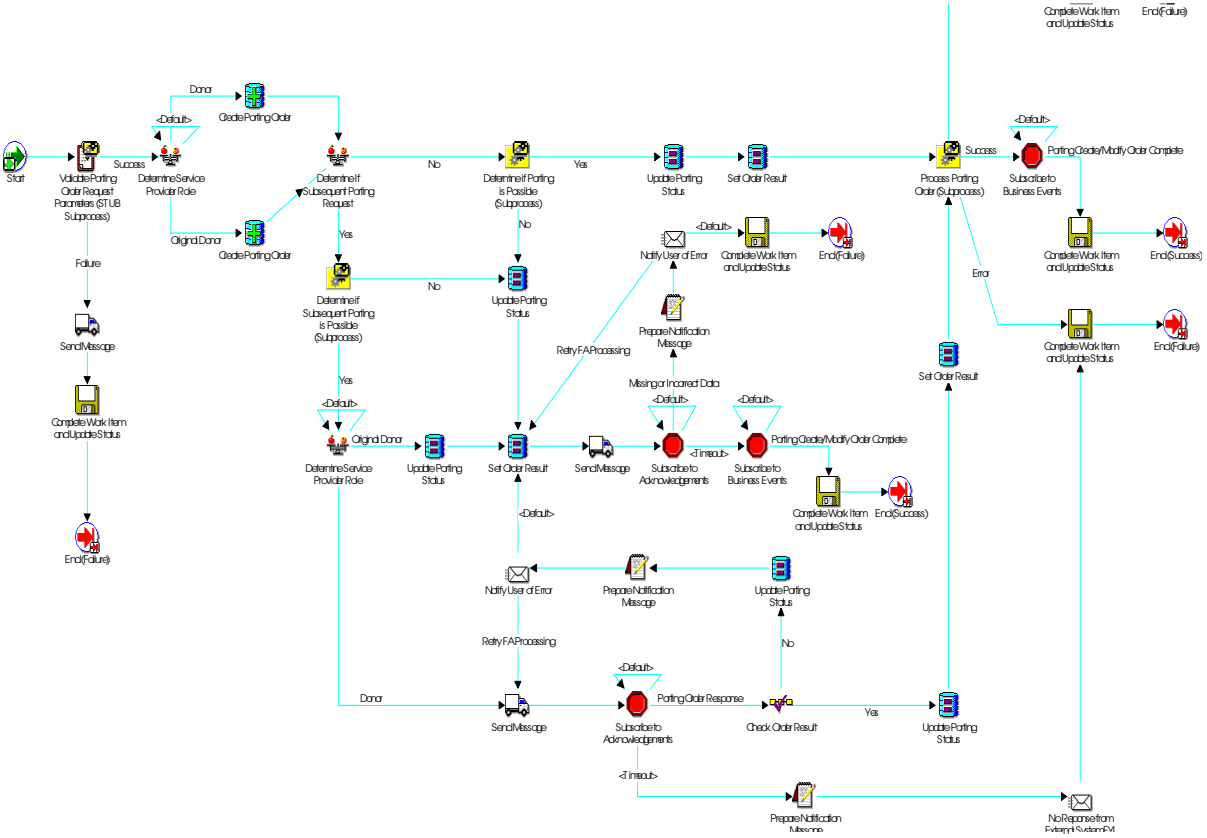
Internal Name:

PORTING_INQUIRY_FROM_OMS

Description:

New Operator receives porting inquiry from Order Management System. A porting inquiry is launched and a response is obtained from the Donor operator and the response is conveyed to the Order Management System.

B.0.4 Respond To Porting Order Received By Donor Operator



Internal Name:

PORTING_ORDER_FROM_OPERATOR

Description:

A porting Order Request is received from the Recipient operator and the required validations are performed to determine a concurrence or rejection. A concurrence or porting rejection reply is sent back to the Recipient operator.

B.0.5 Charge New Operator For Porting Request

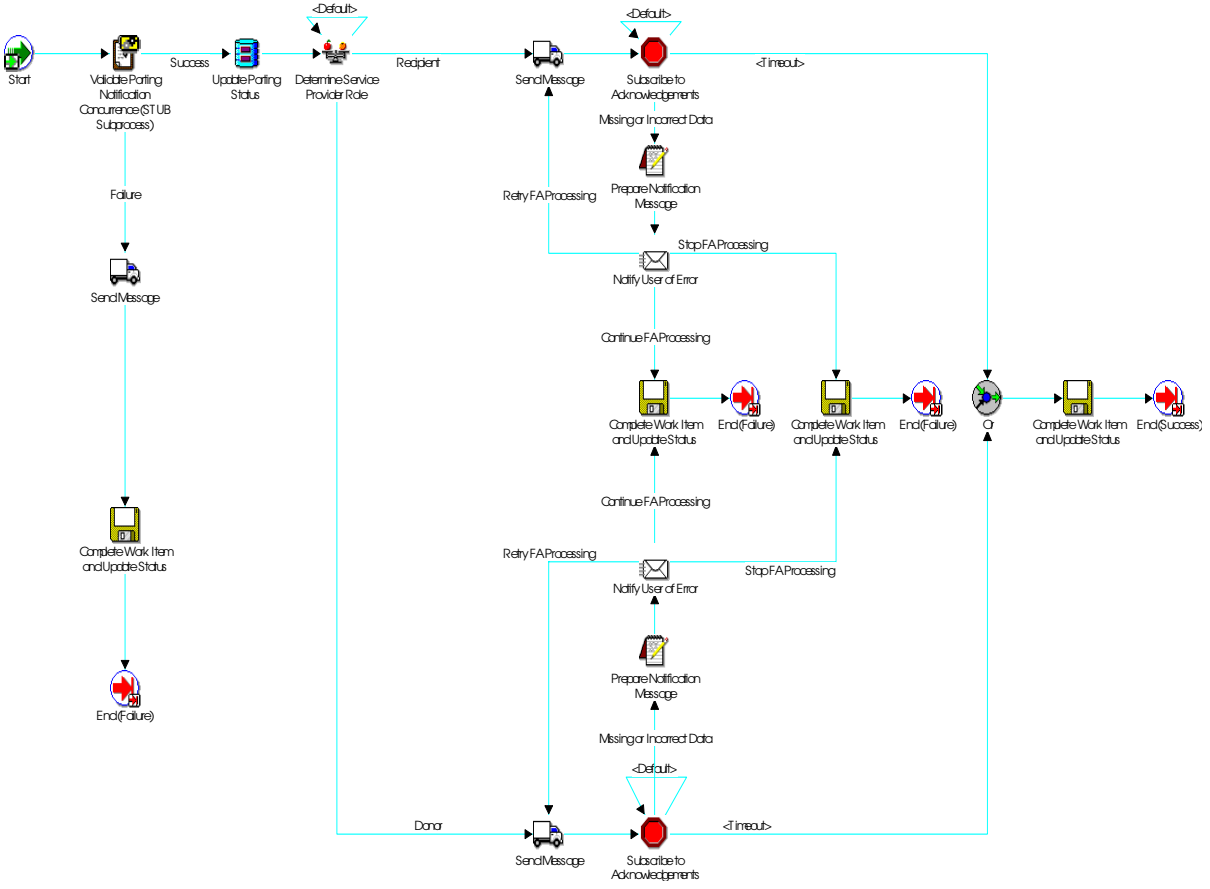
Internal Name:

REC_RECEIVE_CHARGING_NOTIF

Description:

Donor Operator charges recipient operator for the port-out transaction. The cost of the Port Out transaction is added to the invoice of the Recipient operator and the invoice is sent to the Recipient operator.

B.0.6 Porting Notification Concurrence



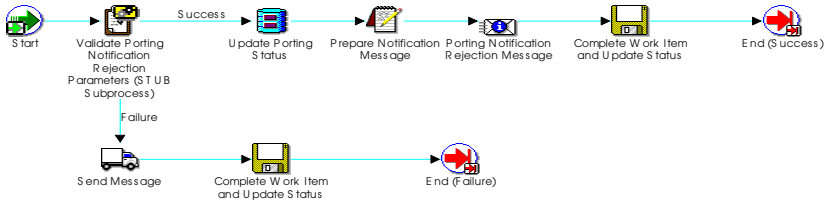
Internal Name:

RECEIVE_CONCURRENCE

Description:

Donor Operator and Recipient Operator receive concurrence notification from NRC for a porting transaction. The Subscription Version status is updated and the operators wait for a network update notification from the NRC.

B.0.7 Reject Porting Request



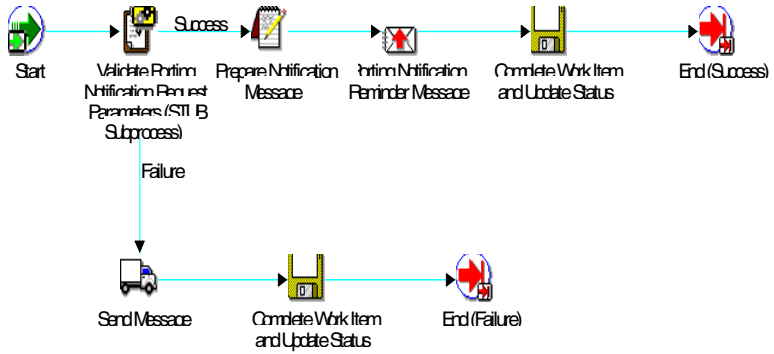
Internal Name:

PORTING_NOTIFICATION_REJECTION

Description:

Donor Operator and Recipient Operator receive rejection notification from NRC for a porting transaction. The Subscription Version status is updated and a notification is sent to the Customer Care System.

B.0.8 Remind Operator For Porting Response



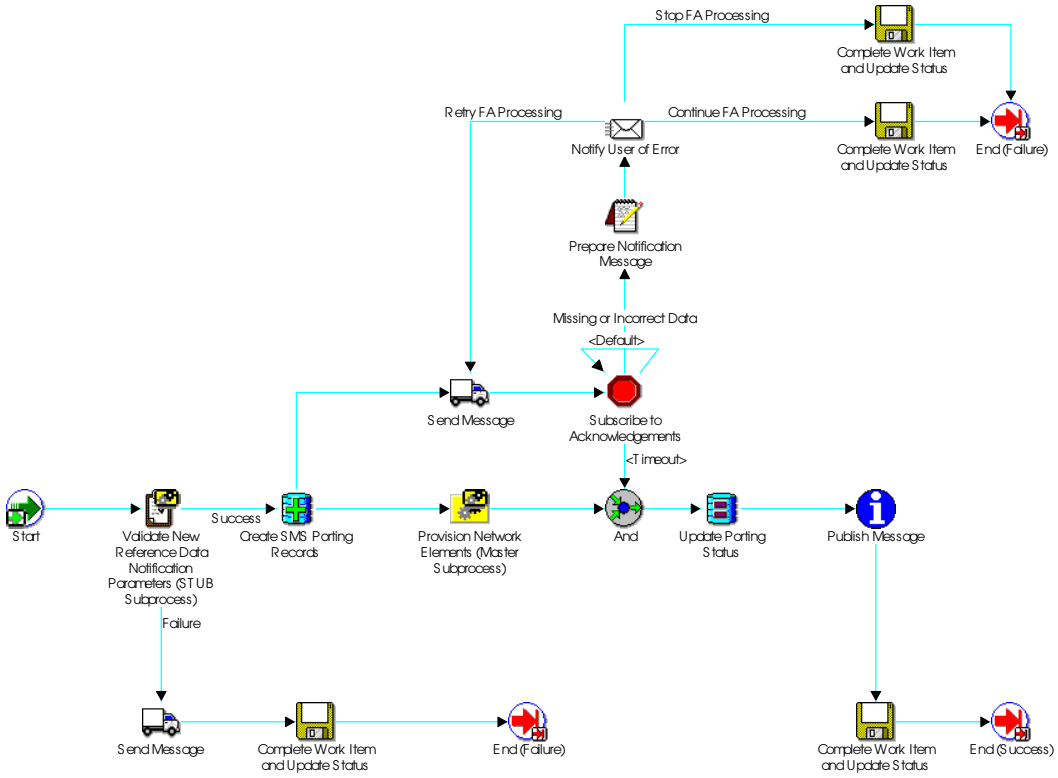
Internal Name:

PORTING_NOTIFICATION_REMINDER

Description:

A Porting Order Inquiry has been sent to the Donor operator and no response has been obtained from the Donor. The Recipient or the NRC reminds the Donor operator of the impending Porting request depending on the implementation. It is assumed that the porting process is initiated by the Recipient operator.

B.0.9 Create Or Modify Ported Number(s)



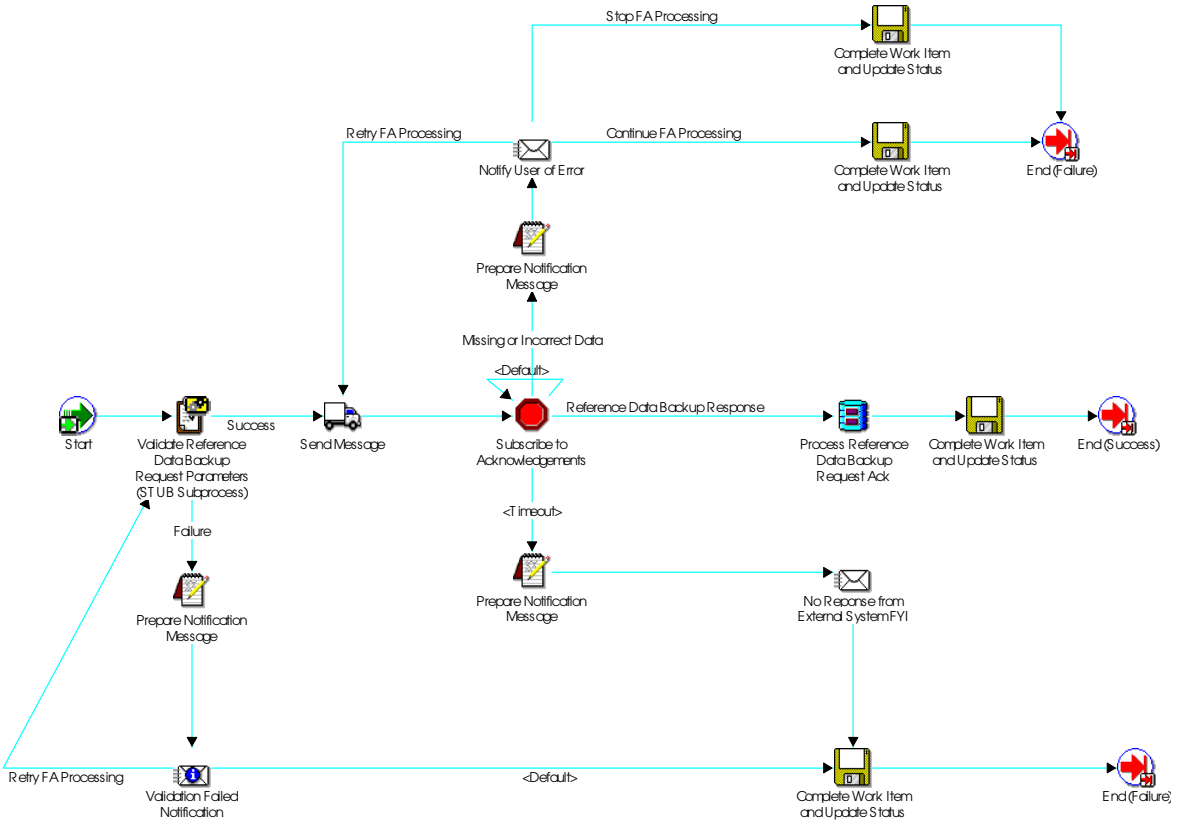
Internal Name:

PROVISION_PORTED_NUMBER

Description:

When a network download arrives from the NRC, the Recipient operator adds the new subscriber to its network while the Donor removes the subscriber from its network. Adding a subscriber may involve adding the telephone number and any of the features or services subscribed to.

B.0.10 Load, Disaster Recovery & Backup Of Local Database



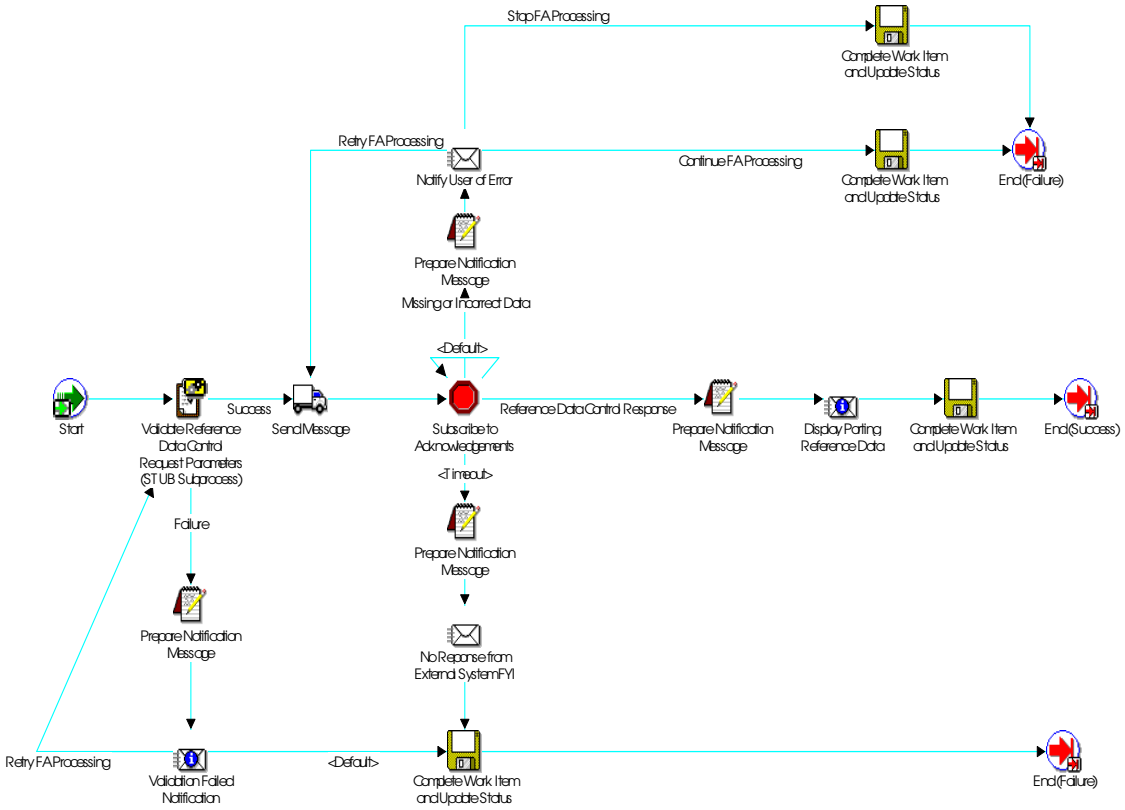
Internal Name:

LOAD_DISASTER_RECOVERY_BACKUP

Description:

Send a network download request to the NRC for synchronizing the local copy of the data with the NRC data and update the local or reference database with data from the administration database.

B.0.11 Query Porting Data



Internal Name:

QUERY_REFERENCE_DATA

Description:

Send a query to the NRC for porting data. This is required for comparing the porting data in the local database with the data in the NRC database. Such comparisons are called Audits.

B.0.12 Modify Porting Request From Order Entry

Internal Name:

MODIFY_PORT_REQ_FROM_OMS

Description:

A Porting Request is pending and the customer wants a modification to the request. One example of such modification is modifying the porting due date. The Subscription Version in the local database is modified and a porting modification request is sent to the NRC. It is up to the business process to allow or not to allow such modifications. An example is, allow any modification prior to twenty four hours of the activation due date but no modifications after that.

B.0.13 Modify Porting Request From Operator

Internal Name:

MODIFY_PORT_REQ_FROM_OPERATOR

Description:

A modification request to the Subscription Version arrives from the NRC or the other operator. Modifications to the subscription version is made and a porting modification acknowledgment is sent back.

B.0.14 Cancel Modify Porting Request From Order Entry

Internal Name:

CANCEL_MODIFY_PORT_FROM_OMS

Description:

A cancel request on a previous modify request arrives from the Order Entry System. The pending modification request is deleted or the modification is rolled back.

B.0.15 Hold Porting Request From Order Entry

Internal Name:

HOLD_PORT_REQ_FROM_OMS

Description:

Customer contacts new operator to place a porting request on hold. The status of the porting request is set to 'HOLD'.

B.0.16 Hold Porting Request From Other Operator

Internal Name:

HOLD_PORT_REQ_FROM_OPERATOR

Description:

A hold on the Porting Request is received from the NRC or the other operator. The status of the porting request is set to 'HOLD'.

B.0.17 Cancel Porting Request From Order Entry

Internal Name:

CANCEL_PORT_REQ_FROM_OMS

Description:

A cancel on the Porting request arrives from the Order Entry System. If a Subscription Version is already created at the NRC, the cancel request will also be sent to the NR. The status of the porting request is set to 'CANCELLED'.

B.0.18 Disconnect Porting Request From Order Entry

Internal Name:
DISC_PORT_REQ_FROM_OMS

Description:
A disconnect request is received from the Order Entry System and a provisioning order is initiated to disconnect the customer from the network.

B.0.19 Cancel Disconnect Porting Request From Order Entry

Internal Name:
CANCEL_DISC_PORT_REQ_FROM_OMS

Description:
A cancel on a previous disconnect request is received from the Order Entry System. The impending disconnect request is deleted.

B.0.20 Number Range Split Request From Order Entry

Internal Name:
NUMBER_RANGE_SPLIT

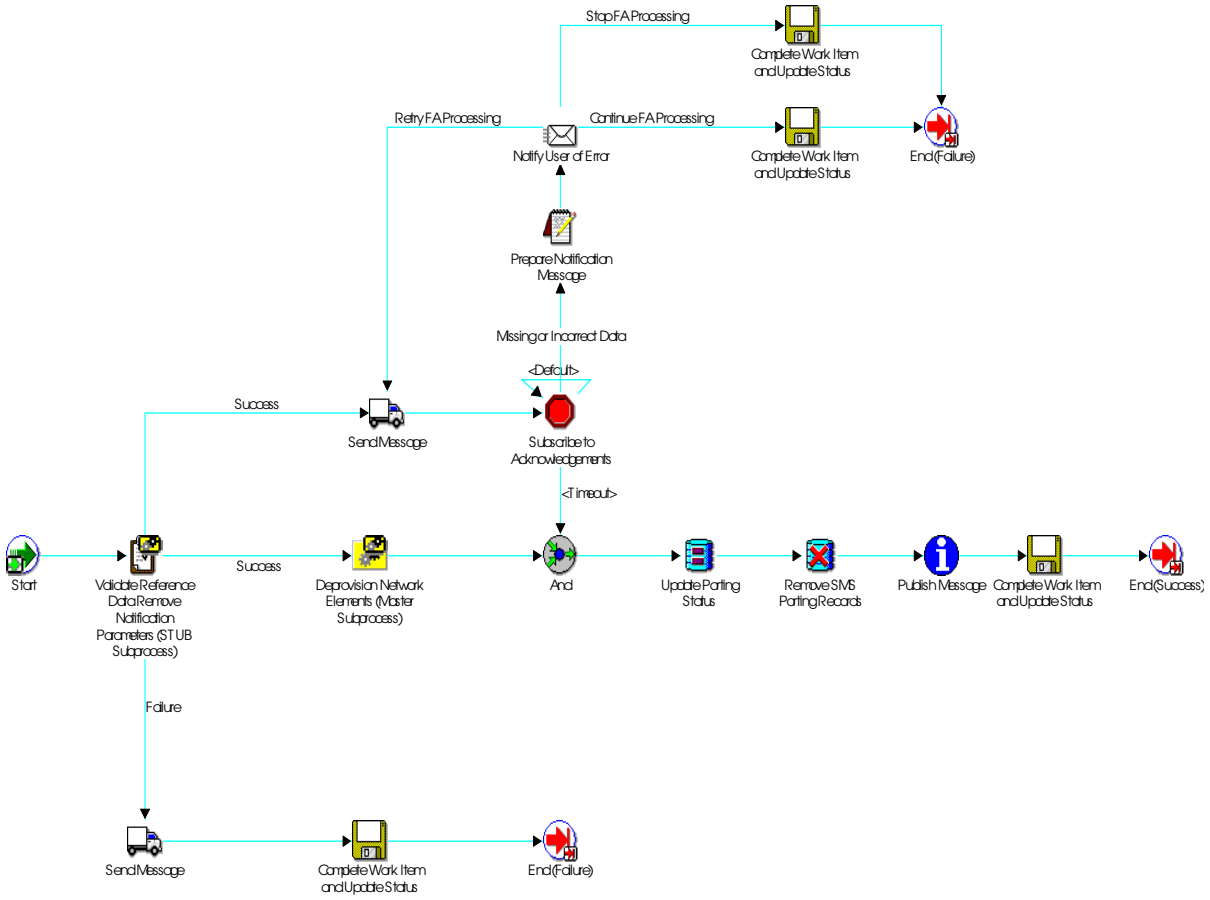
Description:
Number Range Split declared by a regulatory board.

B.0.21 Transfer Number Range Holder From Order Entry

Internal Name:
TRANSFER_NUMBER_RANGE_HOLDER

Description:
Transfer Number Range Holder declared by a regulatory board.

B.0.22 Delete Ported Number(s)



Internal Name:

DELETE_PORTED_NUMBERS.

Description:

Remove porting data from network elements provisioned earlier. This may be in response to a broadcast received from NRC to carry out this activity.

C

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'	'Opp ReferenceIdentification'
'SUBSCRIPTION_TN'	'SubscriptionTN'	'SubscriptionTelephoneNumber'
'STARTING_NUMBER'	'StartingNumber'	'Starting Number'
'ENDING_NUMBER'	'EndingNumber'	'Ending Number'
'PORTING_ID'	'PortingId'	'Porting Identification'
'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'	'ZipCode'	'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'	'Order Reject Explanation'
'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'	'Geographic Area Code'
'PRICE_PER_MINUTE'	'PriceperMinute'	'Price per Minute'
'PRICE_PER_CALL'	'PriceperCall'	'Price per Call'
'PRICE_CODE'	'PriceCode'	'Price Code'
'SERVICE_INFO'	'ServiceInfo'	'Service Information'
'INVOICE_DUE_DATE'	'InvoiceDueDate'	'Invoice Due Date'
'ORDER_PRIORITY'	'OrderPriority'	'Order Priority'
'CHARGING_INFO'	'ChargingInfo'	'Charging Information'
'SENDER_NAME'	'SenderName'	'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'	'ExternalCLASS Address'
'EXT_CLASS_SUBSYSTEM'	'Ext CLASS Subsystem'	'ExteralCLASSSubsystem'
'EXT_ISVM_ADDRESS'	'ExtISVMAddress'	'ExternalISVM Address'
'EXT_ISVM_SUBSYSTEM'	'ExtISVMSubsystem'	'ExternalISVMSubsystem'
'EXT_CNAM_ADDRESS'	'ExtCNAMAddress'	'ExternalCNAM Address'
'EXT_CNAM_SUBSYSTEM'	'Ext CNAM Subsystem'	'ExternalCNAMSubsystem'
'EXT_LIDB_ADDRESS'	'ExtLIDBAddress'	'External LIDB Address'
'EXT_LIDB_SUBSYSTEM'	'ExtLIDBSubsystem'	'External LIDBSubsystem'
'EXT_WSMSC_ADDRESS'	'ExtWSMSCAddress'	'External WSMSC Address'
'EXT_WSMSC_ SUBSYSTEM'	'Ext WSMSC Subsystem'	'ExteralWSMSCSubsystem'
'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'	'ExtRoutingNumber'	'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'	'Order Identification'
'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'	'Number Range Code'
'EFFECTIVE_DATE'	'EffectiveDate'	'Effective Date'
'GEO_INDICATOR'	'GeoIndicator'	'Geographic Indicator'
'MOBILE_INDICATOR'	'MobileIndicator'	'Mobile Indicator'
'POOLED_FLAG'	'Pooled Flag'	'Pooled Flag'
'PORTED_INDICATOR'	'Ported Indicator'	'Set if the Number Range is Open to Portability'
'SERVICE_TYPE'	'Service Type'	'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'	'Primary Flag'	'Primary Flag'
'COUNTRY'	'Country'	'Country'
'DEPARTMENT'	'Department'	'Department'
'INTERNET_ADDRESS'	'Internet Address'	'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'	'Portingactivationstartdate.'
'ACTIVATION_END_ DATE'	'ActivationEndDate'	'Portingactivationcompletedate.'
'AUDIT_TYPE'	'Audit Type'	'Audit Type'
'CLASS_FLAG'	'CLASS Flag'	'CLASS Flag'
'CNAM_FLAG'	'CNAM Flag'	'CNAM Flag'
'FOR_SP_ID'	'For SP Id'	'ForServiceProviderIdentification'
'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'	'NRCName'	'NRC Name'
'MEDIATOR_SP_ID'	'MediatorSPId'	'Mediator Service Provider Identification'
'NRC_SV_ID'	'NRCSVId'	'NRC Subscription Version Identification'
'SUBSCRIPTION_TYPE'	'SubscriptionType'	'Subscription Type'
'SV_SMS_ID'	'SVSMSId'	'Subsription Version SMS Identification'
'SV_SOA_ID'	'SVSOAId'	'Subscription Version SOA Identification'
'ACTIVATION_DUE_ DATE'	'ActivationDueDate'	'Porting Activation Due Date'
'BILLING_ID'	'BillingId'	'Billing Identification'
'BLOCKED_FLAG'	'BlockedFlag'	'Blocked Flag'
'CHANGED_BY_SP_ID'	'ChangedBySPId'	'Changed by Service Provider Identification'
'CONCURRENCE_FLAG'	'ConcurrenceFlag'	'Concurrence Flag'
'CREATED_BY_SP_ID'	'CreatedBySPId'	'Created by service provider identification'
'CUSTOMER_TYPE'	'CustomerType'	'Customer Type'
'DISCONNECT_DUE_ DATE'	'DisconnectDueDate'	'Disconnect Due Date'
'EFFECTIVE_RELEASE_ DUE_DATE'	'Effective Release Due Date'	'Effective Release Due Date'
'LOCKED_FLAG'	'LockedFlag'	'Locked Flag'
'NEW_SP_ AUTHORIZATION_FLAG'	'New SP AuthorizationFlag'	'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'	'PortingTime'	'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 Original Flag'
'STATUS_CHANGE_ CAUSE_CODE'	'Status Change Cause Code'	'StatusChangeCauseCode'
'STATUS_TYPE_CODE'	'StatusTypeCode'	'Status Type Code'
'USER_LOCTN_TYPE'	'User Loctn Type'	'User Location Type'
'USER_LOCTN_VALUE'	'User Loctn Value'	'User Location Value'
'NEW_STATUS_TYPE_ CODE'	'New Status Type Code'	'New Status Type Code'
'OLD_STATUS_TYPE_ CODE'	'Old Status Type Code'	'Old Status Type Code'
'ACTIVE_FLAG'	'Active Flag'	'Active Flag'
'STATUS_NAME'	'Status Name'	'Status Name'
'PHASE_INDICATOR'	'Phase Indicator'	'Phase Indicator'
'DESCRIPTION'	'Description'	'Description'
'DISPLAY_NAME'	'Display Name'	'Display Name'
'LANGUAGE'	'Language'	'Language'
'SOURCE_LANG'	'Source Lang'	'Source Language'
'PRODUCT_TYPE_CODE'	'Product Type Code'	'Product Type Code'
'TIMER_TYPE_CODE'	'Timer Type Code'	'Timer Type Code'
'TIMER_VALUE'	'Timer Value'	'Timer Value'
'UNIT_OF_MEASURE'	'Unit of 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'

D

Number Portability Views

You can use these views to select MLS compliant information from the Number Portability Repository. Selecting data directly from tables is not supported as more entities may be made MLS compliant in the future thereby breaking the existing code.

XNP_GEO_AREAS_VL

The geographic area view provides all the active geographic areas ready for Number Portability.

Name	Null?	Type
ROW_ID		ROWID
GEO_AREA_ID	NOT NULL	NUMBER
GEO_AREA_TYPE_CODE	NOT NULL	VARCHAR2(10)
CODE	NOT NULL	VARCHAR2(20)
ACTIVE_FLAG	NOT NULL	VARCHAR2(1)
CREATED_BY	NOT NULL	NUMBER(15)
CREATION_DATE	NOT NULL	DATE
LAST_UPDATED_BY	NOT NULL	NUMBER(15)
LAST_UPDATE_DATE	NOT NULL	DATE
LAST_UPDATE_LOGIN		NUMBER(15)
DISPLAY_NAME	NOT NULL	VARCHAR2(40)
DESCRIPTION		VARCHAR2(4000)

XNP_MSG_TYPES_VL

The Message type view provides the information interface for Message Types.

Name	Null?	Type
ROW_ID		ROWID
MSG_CODE	NOT NULL	VARCHAR2(20)
MSG_TYPE	NOT NULL	VARCHAR2(10)
STATUS	NOT NULL	VARCHAR2(10)
PRIORITY	NOT NULL	NUMBER
QUEUE_NAME	NOT NULL	VARCHAR2(40)
PROTECTED_FLAG	NOT NULL	VARCHAR2(1)
RESPONSIBILITY_ID		NUMBER
LAST_COMPILED_DATE		DATE
VALIDATE_LOGIC		VARCHAR2(4000)
IN_PROCESS_LOGIC		VARCHAR2(4000)
OUT_PROCESS_LOGIC		VARCHAR2(4000)
DEFAULT_PROCESS_LOGIC		VARCHAR2(4000)
DTD_URL		VARCHAR2(4000)
CREATED_BY	NOT NULL	NUMBER(15)
CREATION_DATE	NOT NULL	DATE
LAST_UPDATED_BY	NOT NULL	NUMBER(15)
LAST_UPDATE_DATE	NOT NULL	DATE
LAST_UPDATE_LOGIN		NUMBER(15)
DISPLAY_NAME	NOT NULL	VARCHAR2(40)
DESCRIPTION		VARCHAR2(4000)

XNP_SV_FE_MAPP_DETAILS_VL

This view provides the fulfillment element used to provision a Number Portability feature for a given SMS entry. It also provides the status of the Provisioning and the type of the fulfillment element used

Name	Null?	Type
SV_SMS_ID	NOT NULL	NUMBER
FEATURE_TYPE	NOT NULL	VARCHAR2(15)
PROVISION_STATUS	NOT NULL	VARCHAR2(20)
PROVISION_STATUS_MEANING	NOT NULL	VARCHAR2(80)
FULFILLMENT_ELEMENT_NAME	NOT NULL	VARCHAR2(40)
DISPLAY_NAME	NOT NULL	VARCHAR2(80)
FULFILLMENT_ELEMENT_TYPE	NOT NULL	VARCHAR2(40)

XNP_SV_ORDER_MAPP_DETAILS_VL

This view provides a mapping of Orders and Work Items that have created or modified a given SOA or SMS porting record. This allows for integration of NP to the SDP Flow Through module

Name	Null?	Type
ORDER_ID	NOT NULL	NUMBER
SV_SOA_ID		NUMBER
SV_SMS_ID		NUMBER
WORKITEM_INSTANCE_ID		NUMBER
EXTERNAL_ORDER_NUMBER	NOT NULL	VARCHAR2(40)
TELEPHONE_NUMBER		VARCHAR2(40)
WORKITEM_ID	NOT NULL	NUMBER
WORKITEM_NAME	NOT NULL	VARCHAR2(40)
WI_TYPE_CODE	NOT NULL	VARCHAR2(40)
WI_DISPLAY_NAME	NOT NULL	VARCHAR2(80)
WI_DESCRIPTION		VARCHAR2(2000)

STATUS_CODE	NOT NULL	VARCHAR2(40)

XNP_SV_SMS_V

This provides all the network data information for the SMS Porting Records

Name	Null?	Type
SV_SMS_ID	NOT NULL	NUMBER
PORTING_ID	NOT NULL	VARCHAR2(80)
ROUTING_NUMBER_ID	NOT NULL	NUMBER
SUBSCRIPTION_TN	NOT NULL	VARCHAR2(20)
SUBSCRIPTION_TYPE	NOT NULL	VARCHAR2(10)
MEDIATOR_SP_ID		NUMBER
PROVISION_SENT_DATE		DATE
PROVISION_DONE_DATE		DATE
CNAM_ADDRESS		VARCHAR2(80)
CNAM_SUBSYSTEM		VARCHAR2(80)
ISVM_ADDRESS		VARCHAR2(80)
ISVM_SUBSYSTEM		VARCHAR2(80)
LIDB_ADDRESS		VARCHAR2(80)
LIDB_SUBSYSTEM		VARCHAR2(80)
CLASS_ADDRESS		VARCHAR2(80)
CLASS_SUBSYSTEM		VARCHAR2(80)
WSMSC_ADDRESS		VARCHAR2(80)
WSMSC_SUBSYSTEM		VARCHAR2(80)
RN_ADDRESS		VARCHAR2(80)
RN_SUBSYSTEM		VARCHAR2(80)
CREATED_BY	NOT NULL	NUMBER(15)
CREATION_DATE	NOT NULL	DATE
LAST_UPDATED_BY	NOT NULL	NUMBER(15)

LAST_UPDATE_DATE	NOT NULL	DATE
LAST_UPDATE_LOGIN		NUMBER(15)
NRC_SP_TYPE		VARCHAR2(20)
NRC_CODE		VARCHAR2(20)
NRC_NAME		VARCHAR2(80)
NRC_TIMEZONE		VARCHAR2(10)
NRC_SERVED_BY_FLAG		VARCHAR2(1)
NRC_ADDRESS_LINE1		VARCHAR2(40)
NRC_CITY		VARCHAR2(40)
NRC_STATE		VARCHAR2(40)
NRC_ZIP_CODE		VARCHAR2(20)
NRC_COUNTRY		VARCHAR2(40)
NRC_PHONE		VARCHAR2(20)
NRC_ADDRESS_LINE2		VARCHAR2(40)
NRC_MOBILE		VARCHAR2(20)
NRC_FAX		VARCHAR2(20)
NRC_PAGER		VARCHAR2(20)
NRC_PAGER_PIN		VARCHAR2(80)
NRC_EMAIL		VARCHAR2(80)
NRC_INTERNET		VARCHAR2(40)
NRC_DEPT		VARCHAR2(80)
ROUTING_REF		VARCHAR2(80)
INTERCONNECT_TYPE		VARCHAR2(10)
ROUTING_SP_ID		NUMBER
ROUTING_NUMBER		VARCHAR2(20)
ROUTING_STATUS		VARCHAR2(10)
REC_SP_ID		NUMBER
REC_SP_TYPE		VARCHAR2(20)
REC_CODE		VARCHAR2(20)

REC_NAME		VARCHAR2(80)
REC_TIMEZONE		VARCHAR2(10)
REC_SERVED_BY_FLAG		VARCHAR2(1)
REC_ADDRESS_LINE1		VARCHAR2(40)
REC_CITY		VARCHAR2(40)
REC_STATE		VARCHAR2(40)
REC_ZIP_CODE		VARCHAR2(20)
REC_COUNTRY		VARCHAR2(40)
REC_PHONE		VARCHAR2(20)
REC_ADDRESS_LINE2		VARCHAR2(40)
REC_MOBILE		VARCHAR2(20)
REC_FAX		VARCHAR2(20)
REC_PAGER		VARCHAR2(20)
REC_PAGER_PIN		VARCHAR2(80)
REC_EMAIL		VARCHAR2(80)
REC_INTERNET		VARCHAR2(40)
REC_DEPT		VARCHAR2(80)

XDP_OE_ORDER_DETAILS_V

View containing the Order entry order details. This view is used for the forms and as a temporary storage for each line item information of an order.

Name	Null?	Type
ORDER_NUMBER	NOT NULL	VARCHAR2(40)
ORDER_VERSION		VARCHAR2(40)
LINE_NUMBER	NOT NULL	NUMBER
LINE_ITEM_NAME	NOT NULL	VARCHAR2(40)
LINE_ITEM_VERSION		VARCHAR2(40)
LINE_ITEM_TYPE		VARCHAR2(20)
IS_WORK_ITEM_FLAG		VARCHAR2(1)

PROVISIONING_REQUIRED_FLAG	NOT NULL	VARCHAR2(1)
ORDER_PROVISIONING_DATE	NOT NULL	DATE
LINE_PROVISIONING_DATE		DATE
PROVISIONING_SEQUENCE		NUMBER
ORDER_TYPE		VARCHAR2(40)
ORDER_ACTION		VARCHAR2(30)
LINE_ITEM_ACTION		VARCHAR2(30)
ORDER_SOURCE		VARCHAR2(40)
ORDER_PRIORITY		NUMBER
LINE_PRIORITY		NUMBER
ORDER_STATUS		VARCHAR2(40)
LINE_STATUS		VARCHAR2(40)
SDP_ORDER_ID		NUMBER
ORDER_DUE_DATE		DATE
LINE_DUE_DATE		DATE
ORDER_CUSTOMER_REQUIRED_DATE		DATE
LINE_CUSTOMER_REQUIRED_DATE		DATE
CUSTOMER_NAME		VARCHAR2(40)
CUSTOMER_ID		NUMBER
ORG_ID		NUMBER
SERVICE_PROVIDER_ID		NUMBER
TELEPHONE_NUMBER		VARCHAR2(40)
RELATED_ORDER_ID		NUMBER
SP_ORDER_NUMBER		VARCHAR2(80)
SP_USERID		NUMBER(15)
ORDER_REF_NAME		VARCHAR2(80)
ORDER_REF_VALUE		VARCHAR2(300)
STARTING_NUMBER		VARCHAR2(80)
ENDING_NUMBER		VARCHAR2(80)

XDP_ORDER_DETAILS_V

View containing the order details. This view should be used by the users to query the order and work item information for any given porting id. The porting id should be first stored in this table in the column as ORDER_REF_NAME = 'PORTING_ID' and the value in ORDER_REF_VALUE.

In the scenario where the Porting id is received only later and not present at the time of order submission, the user can use the Core API XNP_CORE.SOA_UPDATE_PORTING_ID to update the *object_reference* in XNP_SV_SOA, and *order_ref_name* and *order_ref_value* in the XDP_ORDER_HEADERS.

Name	Null?	Type
ORDER_ID	NOT NULL	NUMBER
ORDER_NUMBER	NOT NULL	VARCHAR2(40)
ORDER_VERSION		VARCHAR2(40)
LINE_ITEM_ID	NOT NULL	NUMBER
LINE_NUMBER	NOT NULL	NUMBER
LINE_ITEM_NAME	NOT NULL	VARCHAR2(40)
LINE_VERSION		VARCHAR2(20)
LINE_ITEM_ACTION_CODE		VARCHAR2(30)
WORKITEM_INSTANCE_ID	NOT NULL	NUMBER
WORKITEM_ID	NOT NULL	NUMBER
WORKITEM_NAME	NOT NULL	VARCHAR2(40)
WORKITEM_VERSION	NOT NULL	VARCHAR2(40)
WI_TYPE_CODE	NOT NULL	VARCHAR2(40)
ORDER_STATUS_CODE	NOT NULL	VARCHAR2(40)
ORDER_STATE	NOT NULL	VARCHAR2(40)
LINE_STATUS_CODE	NOT NULL	VARCHAR2(40)
LINE_STATE	NOT NULL	VARCHAR2(40)
WORKITEM_STATUS_CODE	NOT NULL	VARCHAR2(40)
WORKITEM_STATE	NOT NULL	VARCHAR2(40)
LINE_SEQUENCE	NOT NULL	NUMBER

SEQ_IN_PACKAGE		NUMBER
SERVICE_ID		NUMBER
PACKAGE_ID		NUMBER
ORDER_PROVISIONING_DATE	NOT NULL	DATE
LINE_PROVISIONING_DATE		DATE
ACTUAL_PROVISIONING_DATE		DATE
ORDER_COMPLETION_DATE		DATE
LINE_COMPLETION_DATE		DATE
WORKITEM_COMPLETION_DATE		DATE
ORDER_CANCEL_PROVISIONING_DATE		DATE
ORDER_CANCELED_BY		VARCHAR2(40)
LINE_CANCEL_PROVISIONING_DATE		DATE
LINE_CANCELED_BY		VARCHAR2(40)
ORDER_HOLD_PROVISIONING_DATE		DATE
ORDER_HELD_BY		VARCHAR2(40)
LINE_HOLD_PROVISIONING_DATE		DATE
LINE_HELD_BY		VARCHAR2(40)
ORDER_RESUME_PROVISIONING_DATE		DATE
ORDER_RESUMED_BY		VARCHAR2(40)
LINE_RESUME_PROVISIONING_DATE		DATE
LINE_RESUMED_BY		VARCHAR2(40)
ORDER_DUE_DATE		DATE
ORDER_CUSTOMER_REQUIRED_DATE		DATE
LINE_DUE_DATE		DATE
LINE_CUSTOMER_REQUIRED_DATE		DATE
ORDER_ACTION		VARCHAR2(30)
ORDER_SOURCE		VARCHAR2(40)
CUSTOMER_ID		NUMBER
CUSTOMER_NAME		VARCHAR2(40)

ORG_ID		NUMBER
SERVICE_PROVIDER_ID		NUMBER
TELEPHONE_NUMBER		VARCHAR2(40)
ORDER_PRIORITY		NUMBER
LINE_PRIORITY		NUMBER
RELATED_ORDER_ID		NUMBER
ORDER_TYPE		VARCHAR2(40)
NEXT_ORDER_ID		NUMBER
PREVIOUS_ORDER_ID		NUMBER
ORDER_REF_NAME		VARCHAR2(80)
ORDER_REF_VALUE		VARCHAR2(300)
SP_ORDER_NUMBER		VARCHAR2(80)
SP_USERID		NUMBER(15)
STARTING_NUMBER		VARCHAR2(80)
ENDING_NUMBER		VARCHAR2(80)
USER_WF_PROCESS_NAME		VARCHAR2(40)
WF_ITEM_TYPE		VARCHAR2(8)
WF_ITEM_KEY		VARCHAR2(240)

XNP_SV_SOA_VL

Name	Null?	Type
SV_SOA_ID	NOT NULL	NUMBER
PORTING_ID	NOT NULL	VARCHAR2(80)
SUBSCRIPTION_TN	NOT NULL	VARCHAR2(20)
SUBSCRIPTION_TYPE	NOT NULL	VARCHAR2(10)
DONOR_SP_ID	NOT NULL	NUMBER
RECIPIENT_SP_ID	NOT NULL	NUMBER
STATUS_TYPE_CODE	NOT NULL	VARCHAR2(20)

PTO_FLAG	NOT NULL	VARCHAR2(1)
CREATED_BY_SP_ID	NOT NULL	NUMBER
CHANGED_BY_SP_ID		NUMBER
MEDIATOR_SP_ID		NUMBER
ROUTING_NUMBER_ID		NUMBER
PRE_SPLIT_SUBSCRIPTION_TN		VARCHAR2(20)
PREV_STATUS_TYPE_CODE		VARCHAR2(20)
CNAM_ADDRESS		VARCHAR2(80)
CNAM_SUBSYSTEM		VARCHAR2(80)
ISVM_ADDRESS		VARCHAR2(80)
ISVM_SUBSYSTEM		VARCHAR2(80)
LIDB_ADDRESS		VARCHAR2(80)
LIDB_SUBSYSTEM		VARCHAR2(80)
CLASS_ADDRESS		VARCHAR2(80)
CLASS_SUBSYSTEM		VARCHAR2(80)
WSMSC_ADDRESS		VARCHAR2(80)
WSMSC_SUBSYSTEM		VARCHAR2(80)
RN_ADDRESS		VARCHAR2(80)
RN_SUBSYSTEM		VARCHAR2(80)
ACTIVATION_DUE_DATE		DATE
NEW_SP_DUE_DATE		DATE
OLD_SP_DUE_DATE		DATE
OLD_SP_CUTOFF_DUE_DATE		DATE
EFFECTIVE_RELEASE_DUE_DATE		DATE
NUMBER_RETURNED_DUE_DATE		DATE
DISCONNECT_DUE_DATE		DATE
INVOICE_DUE_DATE		DATE
STATUS_CHANGE_DATE		DATE
CREATED_DATE		DATE

MODIFIED_DATE		DATE
CONCURRENCE_FLAG		VARCHAR2(1)
NEW_SP_AUTHORIZATION_FLAG		VARCHAR2(1)
OLD_SP_AUTHORIZATION_FLAG		VARCHAR2(1)
RETAIN_DIR_INFO_FLAG		VARCHAR2(1)
RETAIN_TN_FLAG		VARCHAR2(1)
BLOCKED_FLAG		VARCHAR2(1)
LOCKED_FLAG		VARCHAR2(1)
CUSTOMER_CONTACT_REQ_FLAG		VARCHAR2(1)
BILLING_ID		NUMBER
USER_LOCTN_VALUE		VARCHAR2(80)
USER_LOCTN_TYPE		VARCHAR2(20)
CHARGING_INFO		VARCHAR2(4000)
ORDER_PRIORITY		VARCHAR2(1)
STATUS_CHANGE_CAUSE_CODE		VARCHAR2(20)
PREORDER_AUTHORIZATION_CODE		VARCHAR2(20)
CUSTOMER_ID		VARCHAR2(80)
CUSTOMER_NAME		VARCHAR2(80)
CUSTOMER_TYPE		VARCHAR2(10)
CONTACT_NAME		VARCHAR2(80)
ADDRESS_LINE1		VARCHAR2(40)
ADDRESS_LINE2		VARCHAR2(40)
CITY		VARCHAR2(40)
STATE		VARCHAR2(40)
ZIP_CODE		VARCHAR2(20)
COUNTRY		VARCHAR2(40)
PHONE		VARCHAR2(20)
MOBILE		VARCHAR2(20)
FAX		VARCHAR2(20)

PAGER		VARCHAR2(20)
PAGER_PIN		VARCHAR2(80)
EMAIL		VARCHAR2(80)
INTERNET_ADDRESS		VARCHAR2(40)
PRICE_CODE		VARCHAR2(20)
PRICE_PER_CALL		NUMBER(15,3)
PRICE_PER_MINUTE		NUMBER(15,3)
CREATED_BY	NOT NULL	NUMBER(15)
CREATION_DATE	NOT NULL	DATE
LAST_UPDATED_BY	NOT NULL	NUMBER(15)
LAST_UPDATE_DATE	NOT NULL	DATE
LAST_UPDATE_LOGIN		NUMBER(15)
STATUS_PHASE	NOT NULL	VARCHAR2(20)
STATUS_DISPLAY_NAME	NOT NULL	VARCHAR2(40)
STATUS_DESC		VARCHAR2(4000)
DON_SP_TYPE	NOT NULL	VARCHAR2(20)
DON_CODE	NOT NULL	VARCHAR2(20)
DON_NAME	NOT NULL	VARCHAR2(80)
DON_TIMEZONE	NOT NULL	VARCHAR2(10)
DON_SERVED_BY_FLAG	NOT NULL	VARCHAR2(1)
DON_ADDRESS_LINE1	NOT NULL	VARCHAR2(40)
DON_CITY	NOT NULL	VARCHAR2(40)
DON_STATE	NOT NULL	VARCHAR2(40)
DON_ZIP_CODE	NOT NULL	VARCHAR2(20)
DON_COUNTRY	NOT NULL	VARCHAR2(40)
DON_PHONE	NOT NULL	VARCHAR2(20)
DON_ADDRESS_LINE2		VARCHAR2(40)
DON_MOBILE		VARCHAR2(20)
DON_FAX		VARCHAR2(20)

DON_PAGER		VARCHAR2(20)
DON_PAGER_PIN		VARCHAR2(80)
DON_EMAIL		VARCHAR2(80)
DON_INTERNET		VARCHAR2(40)
DON_DEPT		VARCHAR2(80)
REC_SP_TYPE	NOT NULL	VARCHAR2(20)
REC_CODE	NOT NULL	VARCHAR2(20)
REC_NAME	NOT NULL	VARCHAR2(80)
REC_TIMEZONE	NOT NULL	VARCHAR2(10)
REC_SERVED_BY_FLAG	NOT NULL	VARCHAR2(1)
REC_ADDRESS_LINE1	NOT NULL	VARCHAR2(40)
REC_CITY	NOT NULL	VARCHAR2(40)
REC_STATE	NOT NULL	VARCHAR2(40)
REC_ZIP_CODE	NOT NULL	VARCHAR2(20)
REC_COUNTRY	NOT NULL	VARCHAR2(40)
REC_PHONE	NOT NULL	VARCHAR2(20)
REC_ADDRESS_LINE2		VARCHAR2(40)
REC_MOBILE		VARCHAR2(20)
REC_FAX		VARCHAR2(20)
REC_PAGER		VARCHAR2(20)
REC_PAGER_PIN		VARCHAR2(80)
REC_EMAIL		VARCHAR2(80)
REC_INTERNET		VARCHAR2(40)
REC_DEPT		VARCHAR2(80)
NRC_SP_TYPE		VARCHAR2(20)
NRC_CODE		VARCHAR2(20)
NRC_NAME		VARCHAR2(80)
NRC_TIMEZONE		VARCHAR2(10)
NRC_SERVED_BY_FLAG		VARCHAR2(1)

NRC_ADDRESS_LINE1		VARCHAR2(40)
NRC_CITY		VARCHAR2(40)
NRC_STATE		VARCHAR2(40)
NRC_ZIP_CODE		VARCHAR2(20)
NRC_COUNTRY		VARCHAR2(40)
NRC_PHONE		VARCHAR2(20)
NRC_ADDRESS_LINE2		VARCHAR2(40)
NRC_MOBILE		VARCHAR2(20)
NRC_FAX		VARCHAR2(20)
NRC_PAGER		VARCHAR2(20)
NRC_PAGER_PIN		VARCHAR2(80)
NRC_EMAIL		VARCHAR2(80)
NRC_INTERNET		VARCHAR2(40)
NRC_DEPT		VARCHAR2(80)
OWN_SP_ID	NOT NULL	NUMBER
OWN_SP_TYPE	NOT NULL	VARCHAR2(20)
OWN_CODE	NOT NULL	VARCHAR2(20)
OWN_NAME	NOT NULL	VARCHAR2(80)
OWN_TIMEZONE	NOT NULL	VARCHAR2(10)
OWN_SERVED_BY_FLAG	NOT NULL	VARCHAR2(1)
OWN_ADDRESS_LINE1	NOT NULL	VARCHAR2(40)
OWN_CITY	NOT NULL	VARCHAR2(40)
OWN_STATE	NOT NULL	VARCHAR2(40)
OWN_ZIP_CODE	NOT NULL	VARCHAR2(20)
OWN_COUNTRY	NOT NULL	VARCHAR2(40)
OWN_PHONE	NOT NULL	VARCHAR2(20)
OWN_ADDRESS_LINE2		VARCHAR2(40)
OWN_MOBILE		VARCHAR2(20)
OWN_FAX		VARCHAR2(20)

OWN_PAGER		VARCHAR2(20)
OWN_PAGER_PIN		VARCHAR2(80)
OWN_EMAIL		VARCHAR2(80)
OWN_INTERNET		VARCHAR2(40)
OWN_DEPT		VARCHAR2(80)
ROUTING_REF		VARCHAR2(80)
INTERCONNECT_TYPE		VARCHAR2(10)
ROUTING_SP_ID		NUMBER
ROUTING_NUMBER		VARCHAR2(20)
ROUTING_STATUS		VARCHAR2(10)
COMMENTS		VARCHAR2(4000)
NOTES		VARCHAR2(4000)

XNP_SV_STATUS_TYPES_VL

Name	Null?	Type
ROW_ID		ROWID
STATUS_TYPE_CODE	NOT NULL	VARCHAR2(20)
PHASE_INDICATOR	NOT NULL	VARCHAR2(20)
ACTIVE_FLAG	NOT NULL	VARCHAR2(1)
INITIAL_FLAG	NOT NULL	VARCHAR2(1)
INITIAL_FLAG_ENFORCE_SEQ	NOT NULL	NUMBER
DISPLAY_SEQUENCE	NOT NULL	NUMBER
CREATED_BY	NOT NULL	NUMBER(15)
CREATION_DATE	NOT NULL	DATE
LAST_UPDATED_BY	NOT NULL	NUMBER(15)
LAST_UPDATE_DATE	NOT NULL	DATE
LAST_UPDATE_LOGIN		NUMBER(15)
DISPLAY_NAME	NOT NULL	VARCHAR2(40)
DESCRIPTION		VARCHAR2(4000)

E

Configuring Adapters

E.1 Adapter Architecture

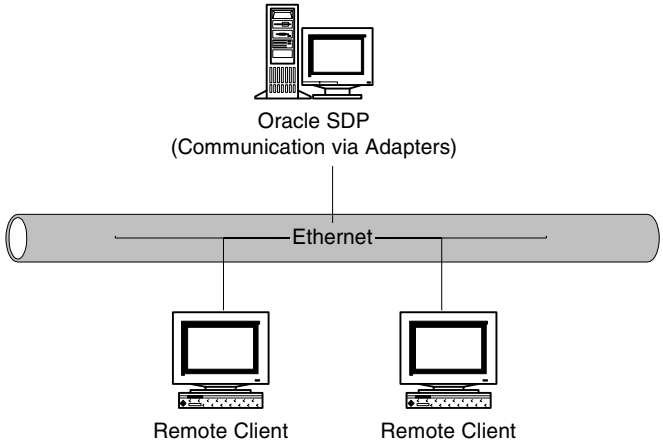
Oracle SDP requires interaction with remote systems & applications to perform application level processing. Remote systems & applications require that any interaction with them happens via well defined interfaces. These interfaces are defined via “Protocol Data Units” (PDUs) for communication over a 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 E–1 Remote Communications via Adapters.

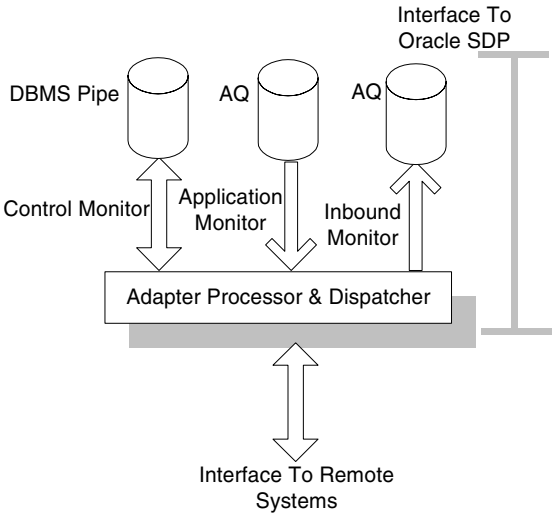


Architecture

The architecture of Oracle SDP adapters can be described as follows:

- Control Message Monitor
- Application Message Monitor
- Inbound Message Monitor

Figure E–2 Adapter Processor & Dispatcher



E.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

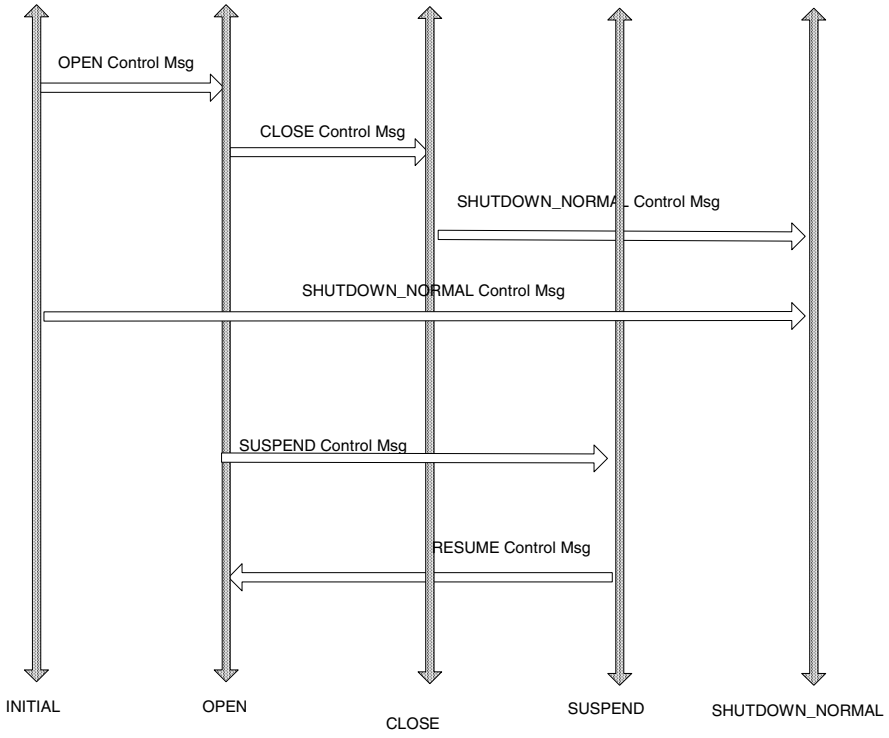
Control messages are messages that are processed by the Control Message Monitor. The various OPERATION types defined for control messages in Oracle SDP adapters is as follows:

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. This control message is another example of an user defined control message and has been implemented in the file adapter.

E.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 E–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"

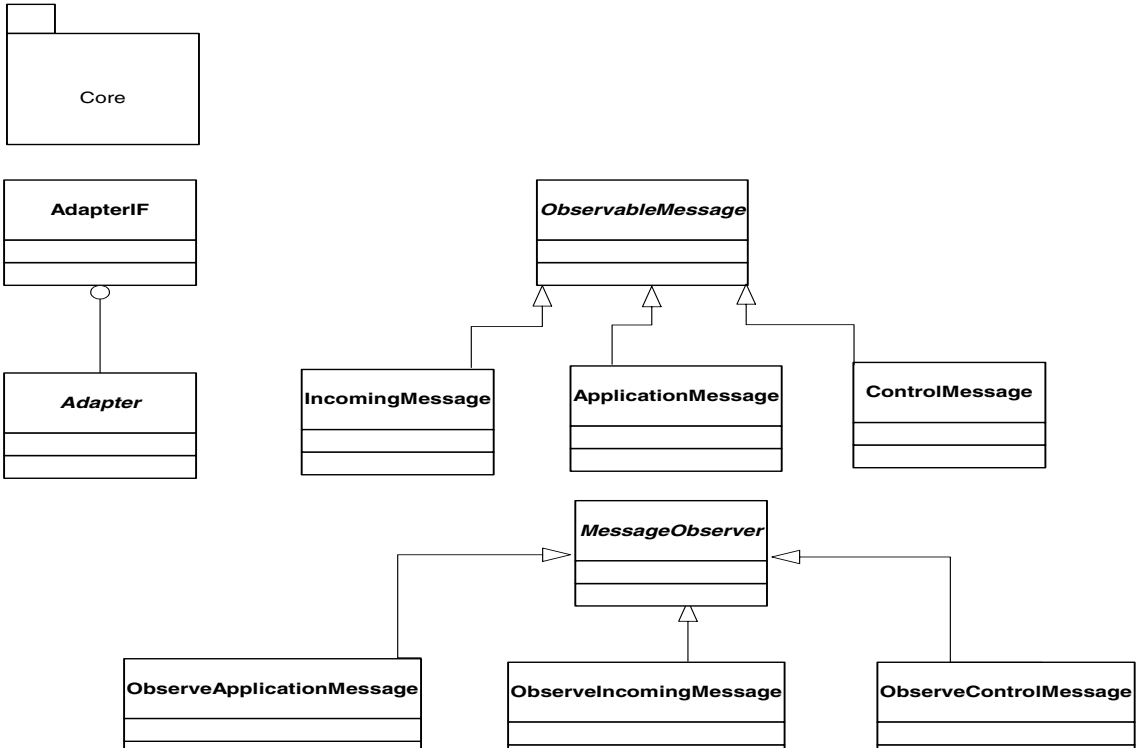
The "core" package defines all the Java classes responsible for the core framework. The classes of the "core" package can be "subclassed" to provide application functionality.

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".

E.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”. 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. This method is invoked by the handleControlMessage method. If the implementation returns a “true” value then everything is considered normal. 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. This method is invoked by the handleControlMessage method. If the implementation returns a “true” value then everything is considered normal. 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. This method is invoked by the handleControlMessage method. If the implementation returns a “true” value then everything is considered normal. 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. This method is invoked by the handleControlMessage method. If the implementation returns a "true" value then everything is considered normal. 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: It invokes createFileName method to create the name of a file if FILE_NAME is not provided. 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.xnp.adapter.impl.file.ScanDirectory;
import oracle.apps.xnp.adapter.impl.file.FileAdapter;
import oracle.apps.xnp.adapter.db.MessageIF;
import oracle.apps.xnp.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.

E.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

E.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.

F

Implementing Number Portability Process Synchronization

Implementing Number Portability Process Synchronization

Oracle Number Portability is a frame work to implement Number Porting business processes. Oracle Provisioning framework is used to execute these business processes as individual NP line items processes which execute a pre-configured NP workitem workflow. The individual workflows are capable of handling the porting process for a single range of telephone numbers (TNs)

In the context of porting multiple ranges of TNs simultaneously, the following requirements need to be satisfied

- The workflows which port the individual TNs ranges, must synchronize with each other i.e., at certain points within the business process, each workflow must wait and proceed only with all the other porting processes have successfully reached the same state. If any of the other porting process had failed in the meanwhile then, all the processes must go to error state. Failure to synchronize after a finite time, must trigger an error process in all workflows and the porting processes must error out with the statuses updated appropriately for all.
- The porting process for all the TNs must be treated as a single order. This would mean, that all the individual workitems must maintain state that it is a part of a whole 'multiple TN range port order' and must be treated likewise by the service providers and the CRDC.
- The messages sent out to the NRC and SPs must indicate that it belongs to a part of a Multiple TN Range port order.
- There should be a provision to send a single message per TN to the legacy system which the TN range porting process interfaces with.
- In a typical Number Portability scenario, one of the Service Providers (SPs) will initiate a request to Port.

Definitions

The following are items to be defined for sychronization:

- SDP Order - The user's entry point to the SDP frame work. The order can be a request to provision network element or NP request with all the necessary parameters.
- SDP Work Item - The SDP order is implemented using smaller functional units called work items
- SDP Line Item - The line item is an abstraction for the workitems and is used to group a bunch of workitems implementing a common business goal.
- Event Manager – The Event Manager is a generic Publish-Subscribe module which registers interest of various subscribers to different event types. The subscriber could be a Translator (in which case the event gets propagated as a new order), Workflow Engine (in which case the event restarts a Workflow which is waiting on an external event) or an API.
- iMessage Studio – The Oracle Provisioning framework for defining messages. Provides support for defining message details, elements, structure, data source, processing logic and test messages.
- Timers - A generic framework of process and activity level timers provided by Oracle NP-OP which can be easily integrated to the NP business process.
- Number Resource Center (NRC) - This is the central system which administers the porting transactions between the Service Providers.
- Donor Service Provider - The Service Provider from which the subscriber wishes to port out from.
- Recipient Service Provider - The Service Provider who gets assigned the portable number range after the porting takes place. In most cases, it is the recipient who initiates the porting transaction. The request is routed to the Donor SP either directly or through the NRC.
- Initiation Phase - The phase in the porting process which spans from submission of the porting request to the state when both operators concur to port.
- Activation Phase - The phase beginning from the end of initiation phase to the beginning of the network provisioning.

Major Features

Multiple TN range porting request requirements are satisfied by having the following features built within the NP-SDP framework

- Provision to group multiple TN range orders at both Donor SP and Recipient SP ends as line items of the same order.
- Provision to synchronize between multiple instances of same workflow with the same synchronization label (defined later).
- Provision to control the 'wait to synchronize' period using Timers.
- Provision to send 1 message per TN to the legacy system by allowing the user to define legacy system messages using the iMessage Studio. The iMessage Studio generated Send procedures could be used to send the legacy system messages.

Order Submission

Oracle NP Order Submission at the Recipient Service Provider End

The Multiple TN range porting order is submitted by the recipient service provider using the SDP's `Process_Order()` API by treating each Number Range Porting request as a line item which directly maps to a workitem workflow. Each line item is given a sequence number and forms part of the same order. The exact syntax for this is provided in the Number Portability Reference Guide. Additionally, the following things need to be incorporated as part of the order submission

- Set the workitem parameter `RANGE_COUNT` for each workitem and value in it will be the Port-in line item number.
- Immediately, after the `Process_Order()` function returns the 'Order id', call the API to register for synchronization between line items passing it the order id (SYNC_REGISTER)
- Ensure that database commit is made only after SYNC_REGISTER is invoked.

Oracle NP Order submission at the Donor Service Provider end

A Multiple Number range Port-Out request comes to the donor service provider as multiple messages each giving information on a single number range. Additionally, the '*comments2*' of the Number Porting Request (NPR) message would contain information indicating that its a part of a whole multiple number range request. The following figure indicates how a NPR message would look like.

At the donor end, an NP Port Out order must be constructed by collating the information in each of these messages and then submitted as one single order. In

this case too, each port out request will be a line item of a multiple number range port out order. The ‘Order Number’ must be uniquely generated. This may be generated by a combination of the ‘installation id’, ‘donor sp. code’ and ‘recipient sp. code’. In order to collate the information, the information in each message is extracted and stored as order and line items in SDP Order Entry tables. When all the messages pertaining to this multiple TN range port out order have been received, the information is collated and submitted as one single order. APIs to collate the order, order line information from these order entry tables and submit already exist and are packaged as part of SDP.

Synchronization at Check Points

On submission of the Multiple TN range porting order, the workitems execute their activities independent of each other until the first synchronization point is reached. In the NP scenario, the synchronization points are at the end of Initiation Phase and Activation phase for both the Donor and Recipient operators.

At these synch points, the workflow waits for the other workflows to successfully reach their synch points. The wait is for a finite interval and after that, the workflows automatically time out an trace a time out path.

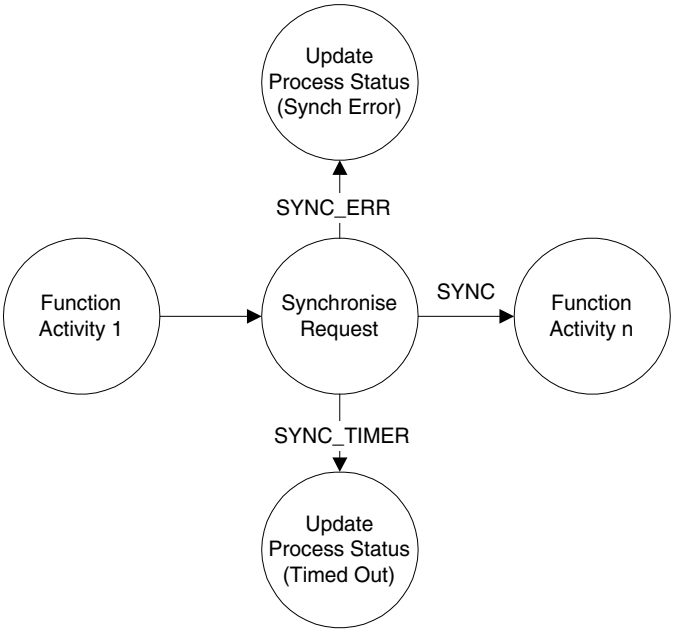
To enable synchronization between multiple workflows, they must register with a synch table prior to start. In the NP scenario, this is done immediately after the Process_Order() API is invoked by calling the SYNC_REGISTER() function. The SYNC_REGISTER() function then generates an unique *synchronization label* using the *order id and line item name* and registers the number of participants against this label. The different synchronizing line items and the number of participants per line item are looked up using the order id. The registration in the case of say, three participants would like the following in the sync table.

Table F–1 Sync. Table - after registration

Sync. Label	No. of participants	Participants not in Sync.	Status
SL1	3	3	Active

Sunny Day Scenario

A typical workflow which synchronizes with the others looks like the following one



When the first workflow waits to synch with the others, a *Synchronize Request) function* is called with the Synch label passed as a parameter. This will decrement the number of participants by one and if the status of the label is still 'Active' then subscribe for the for the following events.

- SYNC with reference id = synch label name + activity label (or id)
- SYNC_ERR with reference id = synch label name + activity label (or id)
- SYNC_TIMER with reference id = synch label name + activity label (or id)

This is with the given synch label concatenated with activity label as the reference id. Now the Synch table will look as follows

Table F–2 Sync. Table - after first synch wait

Sync. Label	No. of participants	Participants not in Sync.	Status
SL1	3	2	Active

Similarly after the second participant synchronizes, the table will now look as follows

Table F–3 Sync. Table - after second synch wait

Sync. Label	No. of participants	Participants not in Sync.	Status
SL1	3	1	Active

When the third workflow reaches the sync point, it notices that it was the last to synchronize. So it publishes a SYNC event with the Synch label + activity id as the reference id and resets the 'Participants not in Sync' to the value in 'No. of participants'. So now the table is ready for the workflows to go ahead and repeat the same scenario at the subsequent synch points

Table F–4 Sync. Table - after all three participants synchronized

Sync. Label	No. of participants	Participants not in Sync.	Status
SL1	3	3	Active

When the SYNC event gets published with the reference id as the synch label, the event manager picks it up and delivers it to both the waiting workflows. Since the synchronized workflows are resumed by the event manager and not by the last synchronizing workflow, all workflows can proceed in parallel once synchronized.

Error Scenario

In case, one of the workflows, the second one say, encounters and error after executing the 'Function Activity 1' then, the first thing it does is to reset the Sync. Table status against this label to 'Error'. So the table would look like the following

Table F–5 Sync. Table - after first workflow synched and second one erred before synching

Sync. Label	No. of partici-pants	Participants not in Sync.	Status
SL1	3	1	Error

In this case, the third workflow Sync Function notices that the Synch. label status is 'Error'. So, after decrement the Parties not in sync, it publishes a SYNC_ERR message with the reference id set to the 'Sync label || activity id' and completes itself along the SYNC_ERR path. The published SYNC_ERR event is picked up by the event manager and delivered to the first workflow which is waiting for the others to synch. So the first workflow too traces the SYNC_ERR path. In the NP scenario, the SYNC_ERR path would contain an activity which updates the status of the porting transaction to indicate that the 'Porting failed' and the status change cause code mentions that the failure was because a peer porting workflow had failed.

Time-out Scenario

In case the first workflow timed out before any of the other two reached the synch point. It would have picked up the SYNC_TIMER event and traced the SYNC_ TIMER path. The processing logic of the SYNC_TIMER event, would reset the status of the Sych label to ERROR. So the table after the processing logic is executed would look like this

Table F–6 Sync. Table - after first workflow synched and second one erred before synching

Sync. Label	**No. of participants**	**Participants not in Sync.**	**Status**
SL1	3	2	Error

When the next two workflows looks up to synch, they notice that they need to trace the SYNC_ERR path. So the treatment will similar to the earlier scenario.

User Procedures Template and Algorithms

This section expands on the information you summarized in the Major Features section. It outlines the procedures users should follow to use the features of your application function. It refers users to other manual sections, such as form and report descriptions, for more detailed information on how to implement those procedures.

Template for Order Submission at Recipient end

> **Note:** Order is constructed as line items, Process_Order is invoked and then Sync. Register is done.

```
CREATE OR REPLACE PROCEDURE OMS_ORDER
IS
 l_error_code      NUMBER := 0;
 l_error_message VARCHAR(2000);
 l_order_header XDP_TYPES.ORDER_HEADER;
 l_order_parameter_list XDP_TYPES.ORDER_PARAMETER_LIST;
 l_order_line_list XDP_TYPES.ORDER_LINE_LIST ;
 l_line_param_list XDP_TYPES.LINE_PARAM_LIST ;
 l_order_id NUMBER := 0;
 l_porting_id NUMBER := 0;
 l_porting_date date := SYSDATE;
 l_can_porting_date varchar2(40) := NULL;

BEGIN
                        -- Order Level Information
 -- Generate an Order Number
 SELECT XDP_ORDER_ID_SEQ.NEXTVAL INTO l_ORDER_ID FROM DUAL;

      -- The following is an example to generate the order no.
      -- In reality the order number must have a more meaning
-- value by concatenating the starting and ending number or
-- by extracting other elements from the received message
 l_order_header.order_number := 'XNP_'||to_char(l_ORDER_ID);
 l_order_header.order_version := '1';

-- Set the SYNC_REQD_FlAG as TRUE

-- Information on Order Line Items
/* Treat each number range as a order line item which maps
 * directly to a workitem workflow. Set the values for each
```

```
 * parameter required by the workitem and then call process order()
 */

-- Repeat the following for each line item. Increment the
-- line item count variable 'i' for additional line item
 l_order_line_list(i).line_number := i;
 l_order_line_list(i).provisioning_required_flag := 'Y';
 l_order_line_list(i).provisioning_date := SYSDATE;

 -- Set the line item name as the workitem name
 l_order_line_list(i).line_item_name := 'PORTING_ORDER_FROM_OMS';
  l_order_line_list(i).is_workitem_flag := 'Y';

 -- Set all the parameters needed for the porting inquiry
 l_line_param_list(1).line_number := i;
 l_line_param_list(1).parameter_name := 'STARTING_NUMBER';
 l_line_param_list(1).parameter_value :=  '87442387';
   l_line_param_list(2).line_number := i;
 l_line_param_list(2).parameter_name := 'ENDING_NUMBER';
 l_line_param_list(2).parameter_value :=  '87442387';

 …. Set any additional parameters needed for this workitem ….
  …. Repeat above Order Line Item information for each number  range
  --- Call process order and get the order id
 xdp_interfaces.process_order
 (l_order_header
 ,l_order_parameter_list
 ,l_order_line_list
 ,l_line_param_list
 ,l_order_id   -- returned from process order
 ,l_error_code
 ,l_error_message
 );
-- Call Sync registration API
SYNC_REGISTER(pp_order_id   => l_order_id
             ,po_error_code => l_error_code
             ,po_error_msg  => l_error_message);
IF lv_error_code = 0 THEN
--- Commit Now
   commit;
ELSE
   Do some error handling
END IF

END;
```

Order Submission at the Donor Service Provider side

Algorithm to collate messages of different number ranges and submit as a single order

```
Parse the comments 2 field to get the total messages, mesg count, installation
id.
If (Comments 2 field indicates that its a part of a multiple number range) then
Generate an unique order number from message elements
If ( entry doesn't exist in xdp_oe_order_headers for this Order number)
then
    Create an entry with this order no.,
    provisioning_date = SYSDATE
End if; // oe entry doesn't exist
Create an Order Line item with
    order no = < generated order no>
    line no = <message count in the comments field>
    line item name = 'PORTING_ORDER_FROM_OPERATOR'
    provisioning_reqd_flag = 'Y'
    is_workitem_flag = 'Y'
Call a procedure to derive each tag value pair in the message   and construct it
as a workitem parameter.
For each message element
    Set order oe line item param count = loop count
    Create order oe Line item param name = message tag name
    Create order oe line item param value = value
End For;
If (the total Line item count for this order no.
      in the order_oe_lines table = 'total messages' in the comments2 field)
then
     Call xdp_submit_oe_order() and get the order id
     Call SYNC_REGISTER(orderid)  ;
end if;
commit;
end if;//(Comments 2 field indicates that its a part of a multiple number range)
```

Synchronize Request Function

```
Synchronise (
  itemtype    IN  VARCHAR2
 ,itemkey       IN  VARCHAR2
 ,actid         IN  NUMBER
 ,funcmode      IN  VARCHAR2
 ,resultout     OUT VARCHAR2
)
IS
```

```
BEGIN
  Get the SYNC_REQD_FLAG parameter from the SDP Order
  IF (SYNC_REQD_ FLAG is false for this orderid or does not exist) THEN
     complete_activity:SYNC; // return here
  END IF;

  Get the SYNC_LABEL from the SDP Order Line

  SELECT status
        ,max_participants
        ,parties_not_in_sync
  FROM   xnp_sync_registration
  WHERE  sync_label = SYNC_LABEL
  FOR UPDATE OF status
              ,max_participants
              ,parties_not_in_sync;
  IF an error is encountered THEN
    Publish a SYNC_ERR message (msg = SYNC_ERR, reference_id = SYNC_LABEL || to_
char(activityid))
    complete_activity:SYNC_ERR; // return here
  END IF;

  IF 1st Synchronise Request THEN
    Start the Synch Timer
  END IF;

  Decrement the parties_not_synched

  IF (status is not 'ACTIVE') THEN
   Update the parties_not_synched in XNP_SYNC_REGISTRATION with the decremented
value
    Publish a SYNC_ERR message (msg = SYNC_ERR, reference_id = SYNC_LABEL|| to_
char(activityid))
    complete_activity:SYNC_ERR; // return here
  ELSE    // status is ACTIVE

  ()
  IF (parties_not_synched > 0) THEN
      Update the parties_not_synched in XNP_SYNC_REGISTRATION with the
decremented value
      Subscribe for a SYNC message(mesg = SYNC, reference id = SYNC_LABEL||to_
char(activityid));
      Subscribe for a SYNC_ERR message(mesg = SYNC, reference id = SYNC_
LABEL||to_char(activityid));
      Subscribe for a SYNC_TIMER message(mesg = SYNC, reference id = SYNC_
```

```
LABEL||to_char(activityid));
    ELSE
      Update the parties_not_synched in XNP_SYNC_REGISTRATION with the max_
participants value
      Publish a SYNC message (msg = SYNC_ERR, reference_id = SYNC_LABEL||to_
char(activityid))
      complete_activity:SYNC;   // Return Here
    END IF
  END IF
END  // End of Synchronise Request
```

Synchronize Registration Function

```
SYNC_REGISTER (
  pp_order_id   IN  NUMBER
 ,po_error_code OUT NUMBER
 ,po_error_msg  OUT VARCHAR2
)
IS
  CURSOR lv_line_item_cur (cv_order_id IN NUMBER) IS
      SELECT order_id||'-'||line_item_name  sync_label
            ,count(*)                       range_count
      FROM   xdp_order_line_items
      WHERE  order_id = cv_order_id
      GROUP BY order_id||'-'||line_item_name;

    -- Cursor to get all Line Item ID's within an Order
    -- based on the SYNC_LABEL
    CURSOR lv_line_item_id_cur (cv_sync_label IN VARCHAR2) IS
      SELECT line_item_id
      FROM   xdp_order_line_items
      WHERE  order_id||'-'||line_item_name = cv_sync_label;
BEGIN

  Add the SYNC_REQD_FLAG parameter to the SDP Order
  Set the SYNC_REQD_FLAG parameter = 'Y'

  FOR each record IN lv_line_item_cur LOOP
    Set the SYNC_LABEL parameter = SDP Order ID||'-'||Line Item Name
    Set the RANGE_COUNT parameter = Nr of duplicate Line Item Names

    FOR each record IN lv_line_item_id_cur LOOP

      Add SYNC_LABEL as a line item parameter
      Add RANGE_COUNT as a line item parameter
```

```
    END LOOP

    Insert a row into the XNP_SYNC_REGISTRATION table

  END LOOP

END // Sync_Register function
```

Raise Synchronize Error Function

```
Raise_Sync_Error (
  itemtype    IN  VARCHAR2
 ,itemkey       IN  VARCHAR2
 ,actid         IN  NUMBER
 ,funcmode      IN  VARCHAR2
 ,resultout     OUT VARCHAR2
)
IS
BEGIN

  Get the SYNC_LABEL parameter value for the SDP Order Line Item
  Update the XNP_SYNC_REGISTRATION table to a status = 'ERROR'
  Publish a SYNC_ERR message (msg = SYNC_ERR, reference_id = SYNC_LABEL||to_
char(activityid))
  complete_activity:SUCCESS;

END // Raise_Sync_Error function;
```

Reset Synchronize Register Function

```
Reset_Sync_Register (
  pp_sync_label IN  VARCHAR2
 ,po_error_code OUT NUMBER
 ,po_error_msg  OUT VARCHAR2
)
IS
BEGIN

    Update the XNP_SYNC_REGISTRATION table to a status = 'ACTIVE'
    Update the XNP_SYNC_REGISTRATION table to a parties_not_in_sync = max_
participants

END // Reset_Sync_Register function;Installation and Upgrade
```

Workflow Changes

The following changes are required to the SDP workflows

Item Type	Function	PL/SQL Procedure Call	Result Type
NP Standard	Synchronize Request	XNP_ SYNC.SYNCHRONISE	Message Type
	Raise Synchronize Error	XNP_SYNC.RAISE_SYNC_ ERROR	Success/Failure

Item Type	Function	PL/SQL Procedure Call	Result Type
NP Standard	Synchronize Request	XNP_SYNC.SYNCHRONISE	Message Type
	Raise Synchronize Error	XNP_SYNC.RAISE_SYNC_ ERROR	Success/Failure

Seed Data

The following changes are required for the SDP Seed Data:

Lookup Codes

The following Lookup Code/s are required

Type Code	Value	Description
XNP_SYNC_REG_STATUS	ACTIVE	Synchronize Registration is ACTIVE
	ERROR	Synchronize Registration is in ERROR

User Display Message

The following User Display Message/s are required

Name	Application	Message Text
XNP_SYNC_ REQUEST	XNP	Synchronize Request Error for &NUMRANGE $: &ERROR_ TEXT

Parameter Pool

The following seed parameter/s are required:

Display Name	Internal Name	Description
Range Count	RANGE_COUNT	Count of the number of Number Ranges which are being ported together
Sync Label	SYNC_LABEL	Unique label used for synchronization
Sync Required Flag	SYNC_REQD_FLAG	Flag to indicate if synchronization was required across line items

iMessage Studio

The following message/s are required:

Message Code	Type	Queue	Elements	Structure	DataSource	Processing Logic
SYNC	Event	Internal Event Queue	SYNC_ LABEL	SYNC SYNC_ LABEL		NULL;
SYNC_ERR	Event	Internal Event Queue	SYNC_ LABEL	SYNC_ERR SYNC_ LABEL		XNP_ SYNC.PR OCESS_ SYNC_ ERR
SYNC_ TIMER	Timer	Timer Queue	Delay Interval	SYNC_ TIMER DELAY INTERVAL	SELECT '0' delay, '100' interval FROM dual;	XNP_ SYNC.PR OCESS_ SYNC_ TIMER

Packages

The following packages and procedures are Required

XNP_SYNC

The XNP_SYNC package will contain all procedures/function to a synchronization request.

```
PROCEDURE Synchronise (
 itemtype     IN  VARCHAR2
,itemkey      IN  VARCHAR2
,actid     IN  NUMBER
,funcmode     IN  VARCHAR2
,resultout    OUT VARCHAR2);
Description Syncrhonise a Workflow Request
PROCEDURE Sync_Register (
 pp_order_id  IN  NUMBER
,po_error_code OUT NUMBER
,po_error_msg OUT VARCHAR2);
Description Register an order for Synchronisation
PROCEDURE Raise_Sync_Error (
 itemtype     IN  VARCHAR2
,itemkey      IN  VARCHAR2
,actid     IN  NUMBER
,funcmode     IN  VARCHAR2
,resultout    OUT VARCHAR2);
Description Set the status of a Synchronisation Request to ERROR

PROCEDURE Reset_Sync_Register (
 pp_sync_label IN  VARCHAR2
,po_error_code OUT NUMBER
,po_error_msg OUT VARCHAR2);
Description Reset the Sync Registration information
```

XDP_ENGINE

The XDP_ENGINE is to be updated to include the following procedures. request.

```
PROCEDURE Reset_Sync_Registration (
 pp_sync_label IN  VARCHAR2
,po_error_code OUT NUMBER
,po_error_msg OUT VARCHAR2);
Description Reset the Sync Registration information
```

XDP_OE_ORDER

The XDP_OE_ORDER package provides an API interface to create Order Entry order details in SDP.

```
PROCEDURE Insert_Oe_Order (
 p_OE_Order_Header IN  xdp_types.oe_order_header
,p_OE_Order_Parameter_List IN  xdp_types.oe_order_parameter_list
,return_code OUT NUMBER
,Error_Description OUT VARCHAR2);
```

Description Inserts into the XDP_OE_ORDER_HEADERS and XDP_OE_ORDER_PARAMETERS tables.

```
PROCEDURE Insert_Oe_Order_Line (
 p_OE_Order_Line IN  xdp_types.oe_order_line
,p_OE_Order_Detail_List IN  xdp_types.or_order_line_detail_list
,return_code OUT NUMBER
,Error_Description OUT VARCHAR2);
```

Description Inserts an Order Line and the associated Line Parameters into the XDP_OE_ORDER_LINES and XDP_OE_ORDER_LINE_DETAILS tables.

```
PROCEDURE Submit_OE_Order (
 p_OE_Order_Number IN  VARCHAR2
,p_OE_Order_Version IN  VARCHAR2 DEFAULT NULL
,SDP_Order_ID OUT NUMBER
,return_code OUT NUMBER
,Error_Description OUT VARCHAR2);
```

Description Selects the Order information from the XDP OE Order tables and inserts into the XDP Order Tables.

Table List

This should include a list of all new and revised tables associated with this design

Actual descriptions and table additions and changes will be addressed in the Detailed Design.

SYNC REGISTRATION

Table F–7 XNP_SYNC_REGISTRATION

Name	Data type	Null, Key	Length
SYNC_LABEL	VARCHAR2	No - UK	240
MAX_PARTICIPANTS	NUMBER	No	
PARTIES_NOT_IN_SYNC	NUMBER	No	
STATUS	VARCHAR2	No	40
ORDER_ID	NUMBER	No	
SYNC_ID	NUMBER	No - PK	

Sending Synchronized Notifications - An Example

The implementation at a customer can use the Standard Synchronization Activities provided by Oracle Number Portability.

Two types of synchronization are identified:

- Synchronization of notifications

  This means one notification will be sent for the Multiple Range of 'x' flows. The last flow will send the notification. An answer to this notification will also result in starting the other, waiting, flows.

- General synchronized processing

  As a general rule, all the flows belonging to one Porting Request will always have to take the same general direction. If a certain question has to be answered (e.g. Is the message validated?), this question has to be answered in the same way for all the flows.

Business Requirement

For some customers, one porting request (consisting of multiple ranges) must always be seen as one logical entity.

The fact that this – in Oracle – is implemented using multiple Workflow is a technical detail in which the user is not interested.

This means that all the communication with the user needs to be done taking this into account. This communication will primarily be done using Workflow notifications. To make things acceptable for the user – and to hide the technical implementation – only ONE notification can be sent to the user for a Multiple Range for a single question.

This means that if 3 flows do not need manual validation, but the 4th one does, then a notification needs to be sent to the user informing him of the status of the 3 ranges which are OK, but also informing him of the fact that the 4th range needs manual validation. This notification can look as follows:

```
Dear User,

Please perform manual validation on the following Porting Request:
CRDC ID: <CRDC ID>
Range 1: <Number From> - <Number To>
Validation 1 : OK
Validation 2 : OK
                        Reject Code: < … >
Range 2: <Number From> - <Number To>
Validation 1 : OK
Validation 2 : OK
                          Reject Code: < … >
Range 3: <Number From> - <Number To>
Validation 1 : OK
Validation 2 : OK
                          Reject Code: < … >
Range 4: <Number From> - <Number To>
Validation 1 : OK
Validation 2 : NOK – Manual Intervention needed
                          Reject Code: < … >
```

Due to a limitation of the Workflow system, the reject codes will have to be entered on a separate page (using an URL), but all the information needs to be present on the notification itself. This can be done using PL/SQL Documents.

Standard Workflow Activities

The following item type attribute needs to be present in the Workflow:

```
XXBG: The Document ID
```

Internal name: XXBGDOCID. This attribute can remain empty at the start of your workflow.

You can copy this attributes over from the 'XXBG: Library of Synchronization Functions' item type.

The following activities are used:

Question.ico

Update Sync Counter and Check if Last Workflow

This activity determines whether the flow to arrive at this point is the last flow, or 'one of the other' flows.

This activity has two possible *results*:

- Last: The flow to arrive is the last flow
- Others: The flow to arrive is one of the others

This activity can be found in the 'SDP Standard' item type and does not need to be changed before it can be used.

> **Warning:** Do NOT copy this activity to your item type. Dragging it to your flow is sufficient


Timer.ico

Process Synchronize if Last Workflow or Wait

This activity lets the flows wait, until the last flow has sent a notification, and received a response. The response will kick off this activity with the same result as the response to the notification.

This activity can be found in the 'SDP Standard' item type. It needs to be copied over to your item type, and the result type needs to be changed to be the same as the notification which will be sent.


Flxasgn.ico

XXBG: Set the Document ID

If your notification has to include data about all the ranges in the Multiple DNR Porting Request, then you have to include this activity just before sending your notification. Please note that practically all the notifications you send have to include information about multiple DNR's, so you must have very good reasons not to include this function. This function makes sure that you know for which DNR's to include the information.


Ntf_urg.ico

<Your Notification>

This notification is sent once for the entire Porting Request, and should therefore contain information concerning the entire Request (see example above). This information can be shown to the user using PL/SQL documents.

This notification needs to have the same possible results as the waiting activity.

You can use any notification you choose to use, the post-notification function needs to be specified as 'XNP_WF_SYNC.SYNCNOTIF'.

The message you make has to use PL/SQL Documents. The value of your PL/SQL Document needs to be exactly as follows:

```
'plsql:<name of your procedure>/&XXBGDOCID
```

The PL/SQL document used in the example uses the following code. It uses the functionality offered by the XXBG_SYNC_PKG package. The example can be used as a template.

```
PROCEDURE PLSQLDOC
   ( document_id   in varchar2,
     display_type  in varchar2,
     document      in out varchar2,
     document_type in out varchar2 )
IS
   v_OrderID       VARCHAR2(100);
   v_WI_Instance   NUMBER;

BEGIN
   -- First get the OrderID
   XXBG_SYNC_PKG.PARSEDOCID( document_id, v_orderID );
   -- Make the message
   FOR Flows_Rec IN XXBG_SYNC_PKG.SYNCED_FLOWS(v_OrderID) LOOP
      v_WI_Instance := Flows_Rec.WORKITEM_INSTANCE_ID;
      XXBG_SYNC_PKG.PRINTLINE(display_type, document,
             '  Starting Number: ' ||
```

XDP_ENGINE.GET_WORKITEM_PARAM_

```
VALUE(v_WI_Instance, 'STARTING_NUMBER'));
      XXBG_SYNC_PKG.PRINTLINE(display_type, document,
             '  Ending Number  : ' ||
```

XDP_ENGINE.GET_WORKITEM_PARAM_

```
VALUE(v_WI_Instance, 'ENDING_NUMBER'));
      XXBG_SYNC_PKG.PRINTLINE(display_type, document);
   END LOOP;
   -- That is it
   XXBG_SYNC_PKG.PRINTLINE(display_type, document);
   XXBG_SYNC_PKG.PRINTLINE(display_type, document, 'That is it ...');

END;
```

Only one branch (the <default> branch) is allowed to leave from your notification. This branch needs to go the 'Process Synchronize if Last Workflow or Wait' activity, which will then proceed (please note that your notification and your 'Process Synchronize if Last Workflow or Wait' activity need to have the same Result Type).

Example Flow

An example flow can be found in Appendix A. This is a notification which is actually used in the 'XXBG: Donor Initiation and Preparation Phase' item type.

Synchronized Processing

Business Requirement

If some automatic validation, consistency checks or general questions take place in a Porting Request, then all the flows in the same Porting Request always need to give the same answer to the questions to ensure synchronization.

For example: If automatic validation of 'x' topics need to take place, then we will define 'x' item type attributes in the workflow. These attributes will then be filled up with the result of the validations. This done for all the flows separately.

At the end of this validation, one has to take all the results into account to come to a final result. This will have to be a synchronized result, because we will have to wait for the last flow to be able to come to a conclusion.

Standard Workflow Activities

The following activities are used:

Question.ico

Update Sync Counter and Check if Last Workflow

This activity determines whether the flow to arrive at this point is the last flow, or 'one of the other' flows.

This activity has two possible *results*:

- Last: The flow to arrive is the last flow
- Others: The flow to arrive is one of the others

This activity can be found in the 'SDP Standard' item type and does not need to be changed before it can be used.

> **Warning:** Do NOT copy this activity to your item type. Dragging it to your flow is sufficient

Timer.ico

Process Synchronize if Last Workflow or Wait

This activity lets the flows wait, until the last flow processes your process which determines the global result for all the flows.

This activity can be found in the 'SDP Standard' item type. It needs to be copied over to your item type, and the result type needs to be changed to be the same as the possible results your process will generate.


Function.ico

<Your Process>

This is a function you need to write. This function needs to determine the result which all the flows are to follow. Below you can find a template you must use. This template can be found in the XXBG_SYNC_PKG package. The parts you have to change are put in bold:

```
PROCEDURE SYNC
     ( itemtype     in  varchar2,
       itemkey      in  varchar2,
       actid        in  number,
       funcmode     in  varchar2,
       result       out varchar2
     )
  IS
     v_OrderID       NUMBER;
     v_GlobalResult  VARCHAR2(100); -- Assign a DefaultValue to this Variable

  BEGIN
     IF    funcmode = 'RUN' THEN
        -- Get the OrderID associated with this Order
        v_OrderID    := WF_ENGINE.GetItemAttrNumber( itemtype, itemkey, 'ORDER_
ID' );
        -- Start Looping through the Waiting Flows

         FOR All_Flows_Rec IN XXBG_SYNC_PKG.SYNCED_FLOWS( v_OrderID ) LOOP
           -- ADD YOUR CUSTOM CODE HERUNDER to Determine the Result ...
--  You have access to the entire XDP_ORDER_

DETAILS_V view
             --   e.g. : WORKITEM_INSTANCE_ID
             --          WF_ITEM_TYPE
             --          WF_ITEM_KEY
             --          ...

             v_GlobalResult := 'GO';

             -- ADD YOUR CUSTOM CODE ABOVE to Determine the Result ...
```

```
        END LOOP;

        -- Save the Global Result
        XNP_WF_SYNC.Set_Result_Code(itemtype, itemkey, v_GlobalResult);

        -- Return to Workflow
        result := WF_ENGINE.eng_completed || ':';

    ELSIF funcmode = 'CANCEL' THEN
       result := WF_ENGINE.eng_completed;
    ELSE
       result := Null;
    END IF;
```

Return;

```
    EXCEPTION
    WHEN OTHERS THEN
       WF_CORE.CONTEXT('XXBG_SYNC_PKG', 'SYNC', itemtype, itemkey, to_
char(actid), funcmode);
       raise;
  END;
```

Only one branch is allowed to leave from your process. This branch needs to go the 'Process Synchronize if Last Workflow or Wait' activity, which will then proceed.

Your process should not have a Result Type associated with it. It merely passes the Global Result to the 'Process Synchronize if Last Workflow or Wait' activity.

Example Flow

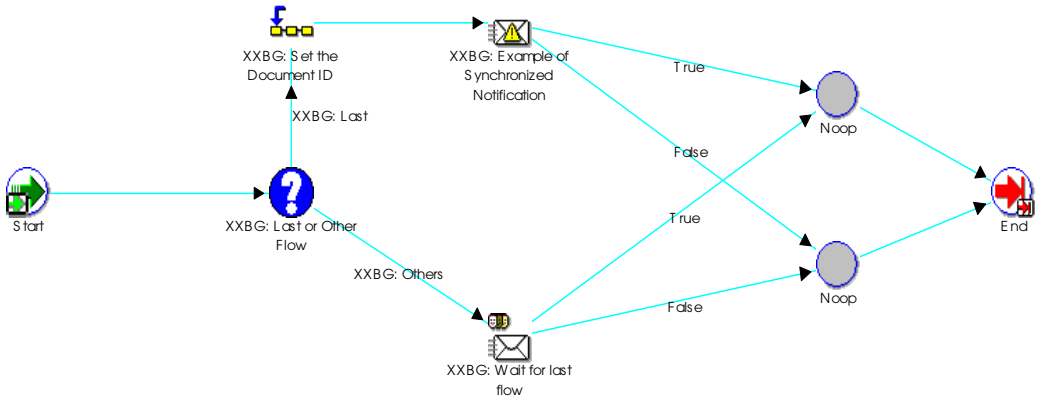
An example flow can be found in Appendix B. This is a synchronization point actually used in the 'XXBG: Donor Activation phase' item type.

The PL/SQL code for the sync point looks as follows:

```
PROCEDURE CONSISTSYNC
     ( itemtype     in  varchar2,
       itemkey      in  varchar2,
       actid        in  number,
       funcmode     in  varchar2,
       result       out varchar2
     )
  IS
     v_OrderID       NUMBER;
     v_GlobalResult  VARCHAR2(100) := XXBG_INIT_PKG.C_ALLOK;
```

```
    v_WI_Instance   NUMBER;
  BEGIN
    IF    funcmode = 'RUN' THEN
       v_OrderID    := WF_ENGINE.GetItemAttrNumber( itemtype, itemkey, 'ORDER_
ID' );

        FOR All_Flows_Rec IN XXBG_SYNC_PKG.SYNCED_FLOWS(v_OrderID) LOOP
            -- Get the WF Instance ID
            v_WI_Instance := All_Flows_Rec.WORKITEM_INSTANCE_
       ID;
            -- Check The Consistency Results HereUnder ...
            v_GlobalResult := XXBG_INIT_PKG.DETERMINE_RESULT(v_GlobalResult, v_
WI_Instance, 'CHECKSTATUS');
            v_GlobalResult := XXBG_INIT_PKG.DETERMINE_RESULT(v_GlobalResult, v_
WI_Instance, 'CHECKFIELDS');
        END LOOP;


 -- Save the Global Result
        XNP_WF_SYNC.Set_Result_Code(itemtype, itemkey, v_GlobalResult);

        -- Return to Workflow
        result := WF_ENGINE.eng_completed || ':';

    ELSIF funcmode = 'CANCEL' THEN
       result := WF_ENGINE.eng_completed;
    ELSE
       result := Null;
    END IF;
    Return;
    EXCEPTION
    WHEN OTHERS THEN
       WF_CORE.CONTEXT('XXBG_ACT_PKG', 'CONSISTSYNC', itemtype, itemkey, to_
char(actid), funcmode);
       raise;
  END;
```

F.0.1 Example Flow for Notification Synchronization



F.0.2 Example Flow for Process Synchronization