

Oracle Enterprise Taxation and Policy Management Self Service

Implementation Guide

Release 1.0.0.0

E36012-01

October 2012

Oracle Enterprise Taxation and Policy Management Self Service Implementation Guide

Release 1.0.0.0

E36012-01

October 2012

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

This software or hardware and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third party content, products and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third party content, products or services.

Contents

Introduction.....	10
Implementation Overview.....	10
Functional Overview.....	10
Technical Overview.....	12
Products.....	12
Solution Components.....	13
Architecture Overview.....	14
Resources.....	15
Self-Service Application Foundation.....	16
Web Service Processing.....	16
Request/Response Message Flow.....	16
Common Message Components.....	17
Transaction Confirmation.....	18
Application Security.....	18
Portal Application Security Model.....	18
Portal Application Secured Pages.....	18
Portal Application Public Pages.....	18
Multi-Language Support.....	19
Selecting the Current Language.....	19
Multi-Language Support for Portal Pages.....	19
Static Text Multi-Language Support.....	19
Dynamic Text Multi-Language Support.....	20
Multi-Language Support for Admin Data Maintenance.....	20
Multi-Language Support for Data Returned by the Revenue Management System.....	20
Multi-Language Support for Custom Content on Portal Pages.....	21
Supporting Additional Languages.....	21
Application Messages and Error Handling.....	21
Application Error Handling Overview.....	22
Revenue Management Back-End System Errors.....	22
Middleware (BPEL) Errors.....	22
Portal Application Errors.....	23
Application Messages Overview.....	23
Revenue Management Back-End System Messages.....	23
Middleware (BPEL) Messages.....	24
Message Translations and Consolidation.....	24
Portal Application Messages.....	25
Displaying Web Services Messages.....	25
Portal Messages Caching.....	25
Portal Application.....	26
Overview.....	26
The Portal Page Template.....	26
Portal Admin Pages.....	26
Portal Functional Pages.....	27
Customizing the Portal Structure and Pages.....	27
Monitoring the Application.....	27
Monitor the Portal Application Using Enterprise Manager.....	27
Monitor Using WebLogic Logs.....	28
Monitor Portal Logs Using Enterprise Manager.....	28
Monitoring Document References.....	28

Settings and Configurations.....	29
System Administration Pages Overview.....	29
Admin Search Page.....	29
Admin Maintenance Page.....	30
Defining System Options.....	31
System Options List.....	31
System Options Maintenance.....	31
Defining Languages.....	32
Language Search.....	32
Language Maintenance.....	32
Defining Messages.....	32
Message Search.....	33
Message Maintenance.....	33
Defining Lookups.....	33
Lookup Search.....	34
Lookup Maintenance.....	34
Lookup Value List.....	34
Lookup Value Maintenance.....	34
Defining Fields.....	35
Field Search.....	35
Field Maintenance.....	35
Defining Validation Rules.....	36
Validation Rule Maintenance.....	36
Validation Rule Maintenance.....	36
Service Requests.....	37
Overview.....	37
Generic Service Requests.....	38
Emailing a Service Request.....	40
Emailing the Request.....	41
Taxpayer Identification Request.....	42
Refund Status Inquiry Request.....	43
Defining Service Requests.....	44
Service Request Search.....	44
Service Request Maintenance.....	44
Service Request: Main.....	44
Service Request: Appearance.....	45
Service Request: Fields.....	45
BPEL Processes.....	47
Implementing Service Requests.....	48
Web Services.....	48
Supported Service Requests.....	48
Messages.....	49
Configuration.....	50
Service Request Config.....	50
Lookup.....	51
Messages.....	52
Validation Rules.....	52
Advanced Navigation.....	53
BPEL DVM Mapping.....	53
OTSS_ServiceRequestType.....	53
OTSS_FieldCodes.....	53
OTSS_MessageNumbers.....	54
How To Enable New Web Service-based Request.....	54
How to Enable a New Email Service Request.....	55

How to Enable a New Identification Request.....	55
How to Enable a New Refund Status Request.....	56
FAQ: Service Request.....	56
One-Time Payments.....	59
Overview.....	59
Payment.....	60
Payment with External Provider.....	62
Defining Payment Destination.....	63
Payment Destination Search.....	63
Payment Destination Maintenance.....	63
Payment Destination - Main.....	63
Payment Destination - Fields.....	63
Defining Payment Provider.....	64
Payment Provider Search.....	64
Payment Provider Maintenance.....	64
Supported Payment Providers.....	65
BPEL Processes.....	65
Integration with Official Payments Corp.....	66
OPC Integration Overview.....	66
OPC Integration Process Flow.....	67
Redirect Payment Process to Official Payments.....	67
Post-back XML Processing.....	70
Payment Report Processing.....	71
Payment Report Process Flow.....	71
Official Payments Integration Configuration.....	72
Customizing the Integration With Official Payments Corporation.....	73
Implementing One Time Payment.....	74
Web Services.....	74
Prepare External Data Plug-in.....	74
Implementing the Payment Class.....	75
Supported Payment Destinations.....	76
Messages.....	76
Configuration.....	76
Lookup.....	76
Payment Destination.....	77
Messages.....	77
BPEL DVM Mapping.....	78
OTSS_PaymentDestination.....	78
OTSS_FieldCodes.....	78
OTSS_PaymentType.....	79
OPC_PaymentType.....	79
OTSS_MessageNumbers.....	80
How to Enable a New Payment Destination.....	80
How to Enable a New Payment Provider.....	80
FAQ: Payment.....	81
Track Your Transaction.....	83
Overview.....	83
Track Your Transaction.....	83
Track Your Transaction BPEL Processes.....	84
Implementing the Track Your Transaction Query.....	85
Web Services.....	85
UI Customization.....	85
Interactive Tax Assistant.....	86
Interactive Tax Assistant Overview.....	86

Web Interviews.....	86
Implementing Interactive Tax Assistant.....	87
Process Flow.....	87
Defining Interview Sets.....	88
Interview Set Search.....	88
Interview Set Maintenance.....	89
Interview Set - Main.....	89
Interview Set - Interviews.....	89
OPA Configuration and Customization.....	89
Advanced Navigation.....	89
SOA/BPEL Integration.....	91
Integration Overview.....	91
Integration Flow Patterns.....	92
Synchronous Flow Without Confirmation ID.....	93
Synchronous Flow With Transaction ID Staging.....	93
Synchronous Flow With Confirmation ID.....	94
Asynchronous Flow With Confirmation ID.....	95
Flows for Official Payments Corporation Integration.....	96
Integration Points.....	97
Integration Solution Flows.....	98
Common Features of All BPEL Processes.....	98
Common Mapping Rules.....	99
Domain Value Maps (DVM).....	101
Setting Configuration Properties.....	103
Setting System Properties.....	103
Module Configurations.....	103
Service Configurations.....	104
Integration With the Revenue Management System.....	110
Confirmation Number Utility Service.....	110
Integration Services.....	111
Adapter Services.....	111
Payment Integration Flow.....	111
Business Details.....	111
Technical Details.....	111
Integration Services.....	112
Web Services.....	112
Prepare Payment Data Integration Flow.....	112
Business Details.....	112
Technical Details.....	112
Integration Services.....	113
Adapter Services.....	113
Web Services.....	113
Database Tables.....	114
Generic Taxpayer Request Integration Flow.....	114
Business Details.....	114
Technical Details.....	114
Integration Services.....	115
Adapter Services.....	115
JMS Queues.....	116
Web Services.....	116
Taxpayer Identification Integration Flow.....	116
Business Details.....	116
Technical Details.....	116
Integration Services.....	117

Web Services.....	117
Refund Status Inquiry Integration Flow.....	117
Business Details.....	117
Technical Details.....	117
Integration Services.....	118
Web Services.....	118
Confirmation Inquiry by ID Integration Flow.....	118
Business Details.....	118
Technical Details.....	118
Integration Services.....	119
Web Services.....	119
Integration With Official Payments Corporation.....	119
Post Payment (Official Payments) Integration Flow.....	119
Business Details.....	119
Technical Details.....	119
Integration Services.....	120
Adapter Services.....	120
JMS Queues.....	121
Web Services.....	121
Process Payment Report (Official Payments) Integration Flow.....	121
Business Details.....	121
Technical Details.....	121
Integration Services.....	121
Adapter Services.....	121
JMS Queues.....	122
Web Services.....	122
Monitoring the Integration.....	122
Monitoring Using WebLogic SOA Enterprise Manager.....	122
Monitoring Using WebLogic Logs.....	122
Error Processing.....	123
Integration Extensibility.....	125
Pre-Transformation Extension Point.....	125
Post-Transformation Extension Point.....	126
Custom Transformations.....	126
Dynamic End-Point URL.....	126
Confirmation Number Utility.....	126
Steps to Implement Extension Points.....	127
Steps to Implement Custom Transformations.....	128
Customizing the Portal Application.....	129
WebCenter Portal Application Override Bundle.....	130
Upload/Download of the Override Application Bundle.....	131
WebCenter Composer and Administration Console.....	131
Administration Console Resources - Pages.....	131
Administration Console Resources - Page Templates.....	132
Administration Console Resources - Skins.....	132
Administration Console Services - Content.....	132
Administration Console Configuration Page.....	132
Portal Page Link Reference.....	133
Portal Page Links From Within the Portal.....	133
Portal Links from External Documents or Sites.....	133
Portal Links for New Portal Pages.....	133
Configuring the Logo and Company Tag Line.....	133
Overriding Portal Application Images.....	134
Configuring the Portal Custom Icons and Links.....	134

Defining Portal Custom Icons.....	135
Defining Portal Custom Links.....	135
Configuring the Portal Copyright Message.....	136
Adding a New Page to the Portal.....	136
Adding Content to a Portal Page.....	136
Adding WebCenter Managed Content to a Portal Page.....	136
Including Images and References in UCM Documents.....	137
Changing Content of a Portal Page.....	137
Changing Page Labels.....	137
Identifying the Bundle ID for a Label on a Page.....	138
Changing Label Text.....	138
Changing the Label Text Value Using Application Override Bundle.....	138
Changing the Label Text Value Using WebCenter Composer.....	139
Customizing Help Content.....	139
Help Component Structure.....	139
Identifying the Bundle ID for a Help Component on a Page.....	139
Changing Help Text.....	140
Changing the Help Text Using Application Override Bundle.....	140
Changing the Help Text Using WebCenter Composer.....	140
Hide/Show a Help Component on a Page.....	141
Changing Portal Page Template.....	141
Changing Portal Skin.....	141
Managing Portal Customization.....	142
Porting Portal Customizations from One Environment to Another.....	142
Retaining Portal Customization After New Product Release Installation.....	142
Moving UCM Content from One Environment to Another.....	142
Appendix A.....	144
WSDL Library.....	144
Generic Service Request - TSTaxpayerServiceRequest.....	144
Taxpayer Identification Request - TSTaxpayerIdentification.....	145
Refund Inquiry Request - TSGetRefundStatus.....	146
One Time Payment - TSOneTimePayment.....	146
Prepare Payment Data - TSPrepareExtPaymentData.....	148
Process Reconciliation Report - TSProcessExtPayReportRecord.....	149
Confirmation Inquiry - TSGetConfirmationInformation.....	150
Common XML Fragments.....	150
Appendix B.....	154
Sample Messages.....	154
GetConfirmationID.....	154
GetRefundStatus.....	155
IdentifyTaxpayer.....	157
OneTimePayment.....	159
PrepareExtPaymentData.....	161
RequestStatusInquiry.....	163
TaxClearanceCertificate.....	164
PrepareExtPaymentData for Official Payments Corporation.....	167
Process Payment Post-back from Official Payments Corporation.....	169
Process Payment Report from Official Payments Corporation.....	171
Appendix C.....	175
Integration with Official Payments Corporation.....	175
Appendix D.....	185
Setup Parameters for Official Payments Co-branding.....	185
Appendix E.....	188

Glossary.....188

Chapter 1

Introduction

Implementation Overview

Oracle Enterprise Taxation and Policy Management Self Service offers a uniform approach across the enterprise to provide taxpayers with on-demand access to valuable information and services. The solution helps a taxation authority quickly provide taxpayers with the ability to make online payments, contact the taxation agency with questions, and request or receive self-guided automated assistance with policies and tax law.

Functional Overview

The self-service solution provides taxpayers with multiple services:

Interactive Tax Assistant (ITA)

- Provides an interactive aid to a taxpayer, helping with common question about tax law-related policies such as filing, credits, deductions, and withholdings.
- Guides the taxpayer through an interactive dialog to determine an answer based on the taxpayer's input.
- Includes a portal page that displays a list of interviews implemented based on Oracle Policy Automation (OPA).

Where Is My Refund?

- Provides taxpayers with information about the current status of their refund processing.
- Implemented based on a Service Request:
 - Taxpayer identity is validated by the service request validation preprocessor.
 - Request is sent to the revenue management system for processing.

- Response is sent to the taxpayer.

Online Payments

- Banking and credit card payment methods.
- Out-of-the-box support for tax returns, collection notices, and payment plans.
- Supports a configurable set of taxpayer identifiers such as name, address, and ID to confirm a taxpayer's identity.
- Interactive taxpayer identity verification prior to payment submission.

Credit card processing with Official Payments Corporation (OPC)

- Out-of-the-box integration with OPC to process credit card payments.
 - The customer is redirected to the Official Payment website to enter credit card information:
 - Supports debit, credit and other types of electronic payments.
 - Validates, approves, rejects electronic charges.
 - Calculates credit card usage fees.
 - Provides payment confirmation.
- Reconciliation:
 - Official Payments provides a file with daily credit card transactions to reconcile against credit card payments stored in the revenue management system. Notifications are created if any discrepancies are found.

Service Requests

- Supports user interaction with the taxation agency on common subjects:
 - Tax law-related questions.
 - Website-related issues.
 - Requests for tax certificates.
 - Changes in personal information.
- Supports multiple communication methods:
 - Request may be emailed to the service team.
 - Request may be sent to a back-end application for processing.
- User receives a confirmation and can check the status of the request online in the confirmation query portal.
- The revenue management system processes the request according to its workflow.
- Service request is configurable:
 - UI is generated based upon request definition.
 - Request definition includes category, fields, processing method.
 - Appearance is controlled by header and footer content.
 - Preview option allows one to view the request during the configuration process.
- Mode of operation:
 - Request can be processed real time.
 - Request can be staged for later response.
- Supports identification mode:
 - Used when user identity should be confirmed before request can be submitted.
 - User's credentials are sent to the back-end application for confirmation.

Technical Overview

Products

Oracle WebCenter represents a combination of the standards-based declarative development of Java Server Faces (JSF), portals, and social networks, and a set of integrated Web 2.0 services to boost end-user productivity.

Oracle WebCenter key components are: WebCenter Framework, WebCenter Social Computing Services, WebCenter Composer, and WebCenter Spaces.

Oracle Universal Content Management is a content management solution. It is integrated with WebCenter via WebCenter Content and provides:

- An infrastructure for managing documents, images, rich media files, and records.
- End-to-end content lifecycle management from creation to archiving.
- Contextual enterprise application integration.

Oracle Application Development Framework (ADF) is Java EE framework that uses Oracle JDeveloper as development environment. ADF integrates with the Oracle SOA and WebCenter Portal frameworks, simplifying the creation of complete composite applications.

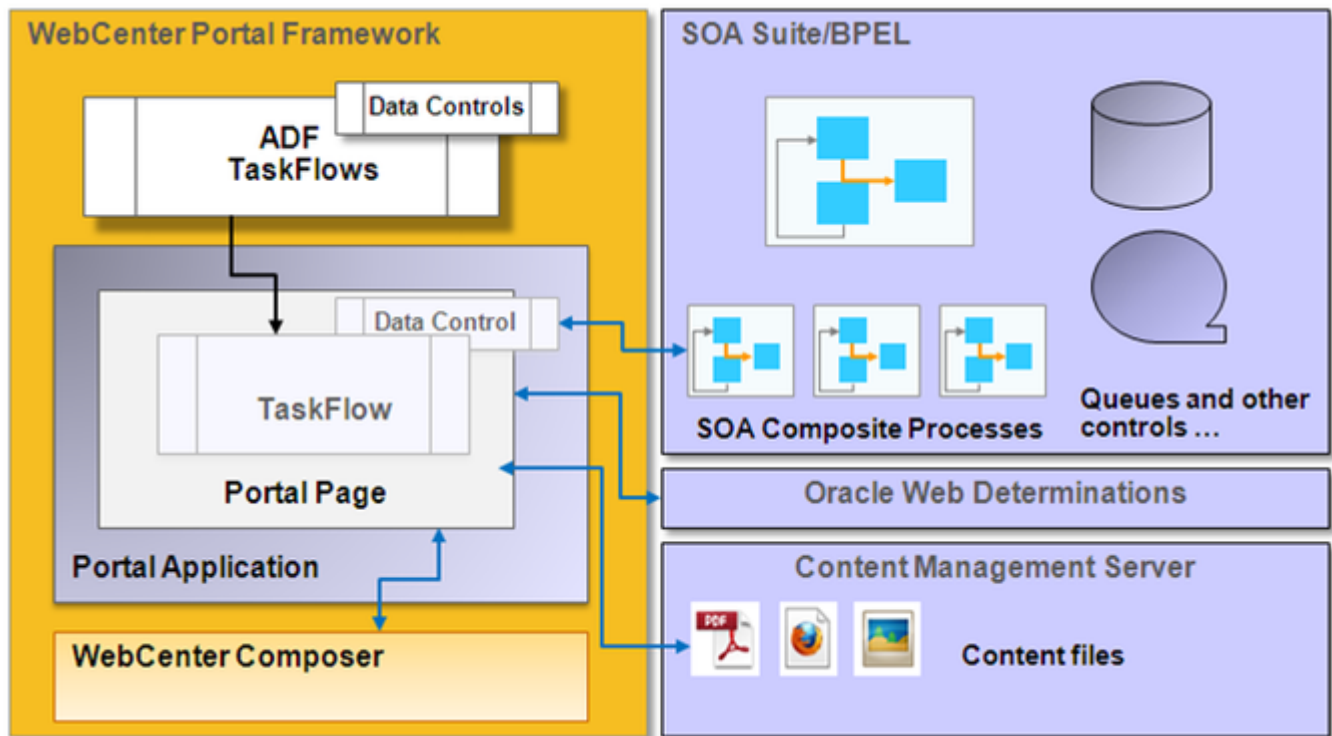
SOA BPEL Process Manager provides a comprehensive, standards-based, and easy-to-use solution for creating, deploying, and managing cross-application business processes with both automated and human workflow steps, all in a service-oriented architecture.

Oracle Policy Automation is a product family designed to automate rules and policies and integrate them into the customer enterprise.

The main technology components of Oracle Policy Automation are:

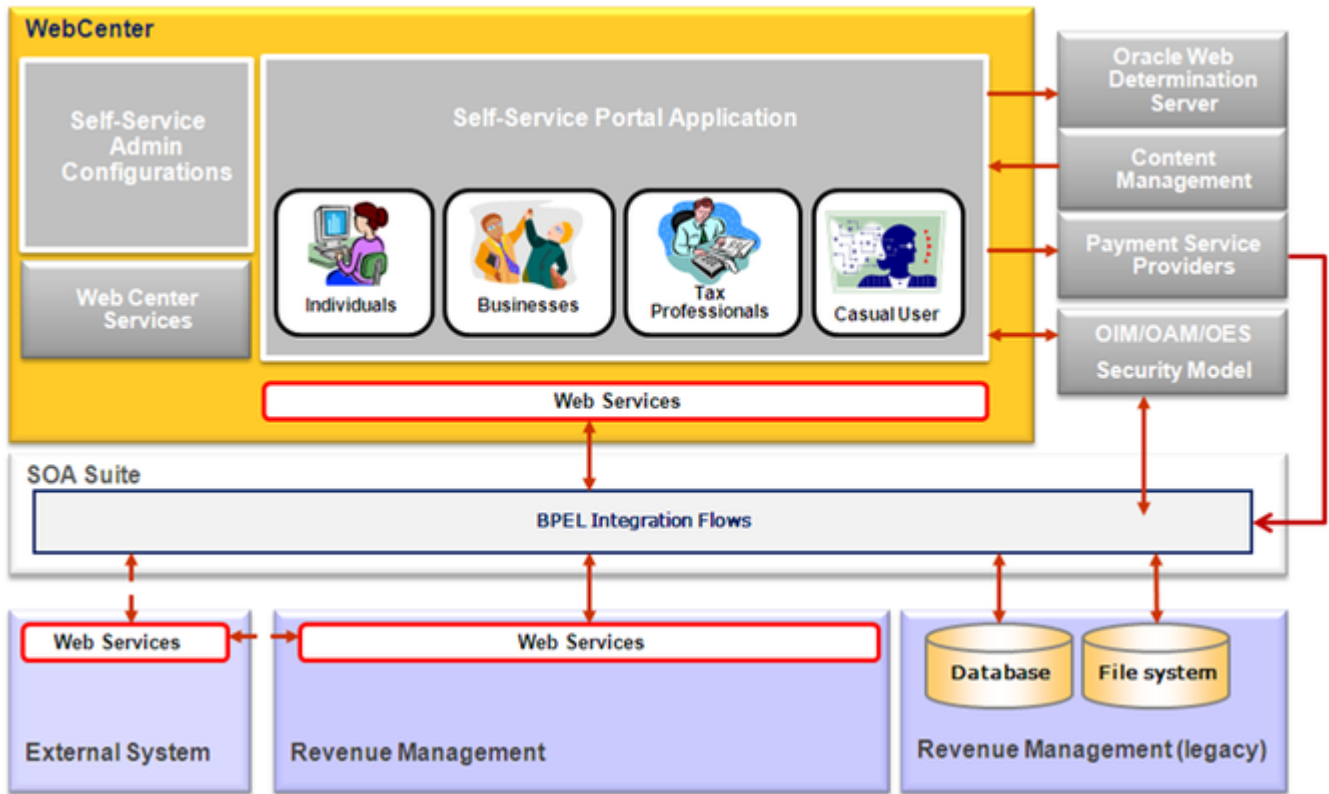
- Oracle Policy Modeling provides a complete natural-language, rule-authoring environment. It is fully integrated with Microsoft Office and includes debugging, regression testing, and what-if analysis for policy changes. Compatible SOA architecture.
- The Oracle Policy Automation suite includes: web service-based interface for remote client applications (Determination Server), Determinations Engine that executes the rulebases at runtime, and an interactive online dialog support (Web Determinations).

Solution Components



- The user interface taskflows are developed using the Application Development Framework (ADF). The taskflow component facilitates data visualization and data interaction through data controls.
- Taskflows are consumed directly on portal pages. The navigation model and the pages are created using WebCenter Portal Framework. Portal application can be extended using WebCenter Composer.
- Content Manager is used as a repository for images and website content, including HTML, pdf, text, and other file types.
- SOA Composites manage communication between the portal application and the revenue management system.

Architecture Overview



On-Line Services is a set of taxpayer-facing portal pages offering the self-service functionality. It can be extended to include additional content pages or otherwise be incorporated into an existing website. The solution supports verification of a taxpayer's identity for the lifetime of a single transaction. This feature allows the casual (not logged in) user to use essential functions, such as a refund status check.

Settings and Configurations is a supplemental module. It maintains a set of control tables that store definitions used for data formatting, translation, and for dynamic UI rendering support. It also includes configuration tables that store business process definitions.

Interactive Tax Assistant is a feature based on integration with Oracle Web Determinations. The base product provides a web interview invocation facility.

External Payment Services. The solution supports delegation of payment services to an external website. The base product includes a pre-built integration with Official Payments Corporation, the largest US online payment services provider.

Content Management is connected to the portal application and serves as the website content repository.

SOA/BPEL Integration Flows facilitate communication between solution components and orchestrate message processing.

The self-service portal is loosely coupled with the back-end system(s). In addition to web services, SOA adapters support file system and direct database interaction. This approach allows use of the product in a heterogeneous environment and accommodates on-going structural changes in the back-end system topology, e.g., a revenue authority introducing a new CRM system.

BPEL processes use Domain Value Maps (DVM) to transform and translate the information maintained in different systems.

Web Services expose the revenue management system's functionality to the online user.

Resources

This Implementation Guide and other self-service application documentation and whitepapers are subject to revision and updating. For the most recent version of this and other Oracle taxation documentation, check the Tax Documentation section of the Oracle Technologies Network (OTN) at <http://www.oracle.com/technetwork/documentation/taxmgmt-154608.html>.

Table 1: Related Documentation and Resources

Resource	Location
Oracle Enterprise Taxation and Policy Management documentation (If using ETPM for back-end data management.)	http://www.oracle.com/technetwork/documentation/taxmgmt-154608.html
Oracle WebCenter documentation	http://docs.oracle.com/cd/E15523_01/webcenter.htm
Oracle Policy Automation documentation	http://www.oracle.com/technetwork/apps-tech/policy-automation/documentation/index.html
Oracle Enterprise Manager (SOA Management) documentation	http://docs.oracle.com/cd/E24628_01/install.121/e24215/soa_overview.htm#GSSOA9844
Oracle Universal Content Management documentation	http://docs.oracle.com/cd/E10316_01/ouc.htm
Oracle Fusion Developers Guide (JDeveloper and Application Development Framework) documentation	http://docs.oracle.com/cd/E23943_01/web.1111/b31974/toc.htm

Chapter 2

Self-Service Application Foundation

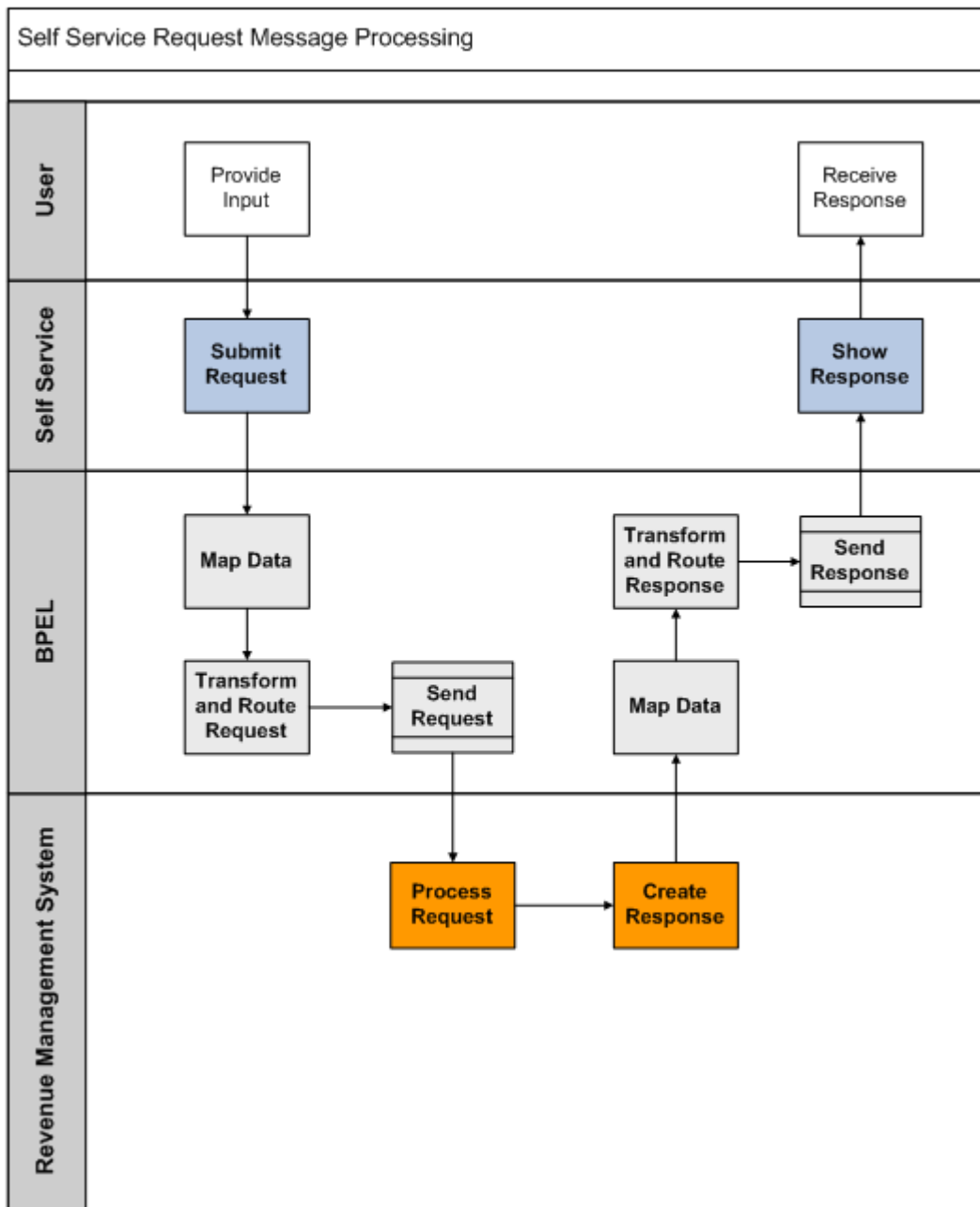
This chapter describes the processes, procedures, and design components that comprise the self-service application framework.

Web Service Processing

Request/Response Message Flow

The self-service application offers two major types of user/revenue authority interaction:

- **Inquiry request.** The user enters a set of criteria, and a web service request is invoked. The information is retrieved based on the input criteria, returned with the web service response, and displayed on the portal page.
- **Service request.** The user is prompted to enter service-related information. The web service request is routed to the revenue management system, where it triggers a business process (e.g., payment creation). The response contains the confirmation message(s) with a reference number.



Common Message Components

All XML messages include the following common fragments:

- **Audit** – Contains the web user details (web user ID, name, email address) and the IP address of the client.
- **Access Keys** – Up to five name/value pairs for business identifiers (for example, taxpayer IDs).
- **Error** – Reserved for the business error messages returned by the revenue management system.
- **Confirmation** – Contains the confirmation ID and message(s).

For additional information, see [Common Features of All BPEL Processes](#) and "Common Components" in the [WSDL Library \(Appendix A\)](#)

Transaction Confirmation

Confirmation numbers are generated in the integration layer and assigned to the <confirmation Id> element. See [Common Features of All BPEL Processes](#) and [Confirmation Number Utility Service](#) for details.

Application Security

- The base product portal application is built from two categories of pages:
 - **Secured pages.** These pages are used for administration purposes and are typically accessed by administrators or implementers.
- **Public pages.** These pages are used by taxpayers that are accessing the self service website.

Portal Application Security Model

The base product portal application is configured with one **Security Role** and one Administrator **User ID**.

The security role that is configured within the base product is **Administrator**.

The user ID and password for that role are provided as part of the installation procedures and can be changed by users as needed.

User ID and password maintenance as well as authentication are managed in the web server. The portal application uses the WebCenter security model, integrated with WebLogic to enforce the authentication of admin users.

Additional users can be defined by implementers in WebLogic if desired.

The portal login page URL is:

```
http://<server>:<port number>/etpmss/faces/oracle/webcenter/portalapp/pages/login.jspx
```

Portal Application Secured Pages

All of the portal pages under the **Settings** and **Configurations** menus are secured. These pages are only accessible after a successful login with a user ID that has the **Administrator** role.

The initial entry into the portal secured pages is the login page which is *not* a secured page. After a successful login, the user is redirected to the home page with a menu bar that includes the admin pages access.

Portal Application Public Pages

Taxpayers who log into the website will be accessing the public pages only. There is no link on the public pages for an administrator login.

The portal functional pages, such as payments or refund status request pages, as well as the home and online services page are all public pages.

The portal home page for taxpayers is:

```
http://<server>:<port number>/etpmss/index.html
```

Multi-Language Support

The base product portal application is built to support multiple languages.

Multi-Language is supported at several levels:

- Text displayed on portal pages.
- Admin data maintenance and the data that is maintained in the portal application.
- Data returned from the revenue management system as a result of the base product-provided web services.
- Custom content in portal pages.

Note: The base product portal application is provided with a default language setup of **English**.

Selecting the Current Language

The base product displays all pages in the default language that is set up in the system (in the **System Options** definitions). The base product portal **Home** page displays the supported languages on the top-right part of the screen. Languages are displayed as Language Code (that was defined in the system Language setting entity).

To switch to a different language, click an available language code hyperlink.

When a new language is selected, the portal page will refresh.

For more information about defining languages and the system default language, see the [Settings and Configurations](#) chapter.

Multi-Language Support for Portal Pages

Two types of text is displayed on the base product portal pages:

- **Static Text**, which includes labels, headers, buttons, page titles, etc.
- **Dynamic Text** is text that is derived from admin data in the system. This includes, for example, the support for Service Request or Payment Destination fields and field level help, as well as Service Request page headers and footers (depending on the request type).

Static Text Multi-Language Support

All static text that is display on the base product portal pages derives its values from **Resource Bundles**.

The values are organized in the resource bundles according to **Resource Bundle IDs**. There are separate resource bundle files for each supported language (identified by a two-character language code in the file name, e.g., `filename_en.xlf`).

The WebCenter application framework automatically selects the value that is displayed on a page based on the values from the resource bundle files and according to the current displayed language.

Some of the multi-language support in the portal application is derived automatically from the ADF framework. For example, the labels on the buttons on the admin data search pages are not defined in the portal application, but are a part of the ADF framework. When the language is switched, ADF automatically changes the label for these buttons to the value corresponding to the new language.

Note:

There should be a separate Application Override Bundle file (**WSSPortalOverrideBundle_<language code>.xlf**) for each supported language so that all modified values can be displayed for the selected language.

For more information about resource bundles and the Application Override Bundle, see the [Customizing the Self Service Portal Application](#) chapter.

Dynamic Text Multi-Language Support

Portal pages that display text and fields controlled by admin data definition support multiple languages as a result of the multi-language support of the admin data management. For example, if a service request page displays a few fields included in the service request definition (which is admin data), the value of text to be displayed on the page will be taken from the appropriate value of that admin data according to the currently displayed language.

The underlying assumption here is that if your portal supports multiple languages—both English and Spanish, for example— all of the admin data that supports multiple languages is populated with the appropriate values for those languages.

Multi-Language Support for Admin Data Maintenance

Portal application admin data includes data that is language insensitive (such as numbers or codes) and data that is language sensitive (such as text values that appear on the page, for example: lookup value descriptions that appear in dropdown lists).

All the language sensitive admin data is stored in language tables according to the language code that they were saved under.

For example if a user is working as an English language user, all the language sensitive data is saved under the language code of "en". If the same user switches to the Spanish language, the language sensitive admin data is saved under a difference language code ("es").

When language sensitive data is presented to the user the only the values for the **Current Language** values are displayed and can be updated.

Important: Admin data language sensitive data has to be maintained for *each* of the supported languages. For example, if the portal supports one additional language in addition to English, when adding a new lookup, the description of the lookup and the description of the values associated with the new lookup have to be maintained once when user is working as an English user, and the second time after user switches to the second supported language.

Multi-Language Support for Data Returned by the Revenue Management System

Data returned to the portal application from a revenue management system, via base product web service calls, can contain many data types.

The base product web services returns message codes from the back-end system for the purpose of displaying data to the portal user. The back-end system can return a set of message codes that can be translated to a different set of message codes by the base product middleware layer (SOA Composites/BPEL processes) that are finally displayed on the portal page.

Since messages are a part of the portal admin data, they should contain the text to be displayed in all the supported languages so that the portal can display the correct text according to the currently selected language.

For more information about message handling, refer to [Application Messages and Error Handling](#)

Multi-Language Support for Custom Content on Portal Pages

Portal pages, whether base product or custom created, can include custom content. Some of the types of content provide multi-language support. One example is UCM content. When including UCM documents on a portal page (using WebCenter Composer, using the Content Presenter component) there are two options:

1. Provide the exact document reference (for example, document name or document id); or
2. Provide the document reference via a resource bundle.

In order to support multi-language for UCM documents, implementers can do the following:

- Create multiple documents for the same content for multiple supported languages. For example, an HTML document that is included on the home page can be created in several languages.
- When including the UCM document on a portal page via WebCenter Composer, implementers should define the reference for that document using a resource bundle for each of the supported languages using the SAME resource bundle id. For example, if the portal supports two languages, English and Spanish then implementers have to define the reference to the UCM document while working in each of these languages.

Each time the document referenced is defined, the same resource bundle id should be used but the value is expected to be different since the documents are likely to be different.

Note:

Referencing a UCM document as a value in a resource bundle can be done by directly providing the **Data Source** and **Data Source Type** parameters to the Content Presenter component through which the UCM document is included on the page.

The **Data Source Type** should be *Single Node* and the **Data Source** should have the expression `UCM#dDocName:<UCM Content id>` entered as a resource bundle value.

For more information about including UCM documents in portal pages, refer to the chapter, [Customizing the Self Service Portal Application](#).

Supporting Additional Languages

Implementers can add new supported languages (in addition to the languages that are supported by the base product) to the portal application. The process of enabling an additional language typically includes the following tasks:

- Defining a new language in the system language settings.
- Adding all the resource bundle keys (from the **Application Override Bundle** template file `WSSPortalOverrideBundle_ReferenceXLF.txt`) to the application override bundle for the new language (`WSSPortalOverrideBundle_<new language code>.xlf`).
- Translating all the values in the new language override bundle.
- Adding new language dependent values in all the admin data, while working in the new language (by switching to that language in the portal).

Application Messages and Error Handling

Application Error Handling Overview

System or application errors can occur in all three layers that make up the base product application: the portal, the middleware, and the revenue management back-end system when web services are invoked.

Note: Errors typically result in messages being displayed on the portal application. For information about the handling of application messages, refer to the [Application Messages Overview](#) section.

Revenue Management Back-End System Errors

The connection between the portal application and the back-end system is made through web services. Data inquiries and updates are performed through web service calls from the middleware (BPEL) layer to the back-end system. Thus, the errors described in this section are all related to the availability of and response provided by the back-end system to web service calls:

- **"Back-end system is not available"** means the system is down or not responding.
- **"Web service call result from back-end system is a SOAP fault"** is typically the result of an application or system error in the back-end system.
- An error can be generated if the web service call to the back-end system doesn't return any message (e.g., neither confirmation nor error messages are returned to the caller).

All of these errors are detected and dealt with in the middleware (BPEL) layer. For more information, refer to the [SOA/BPEL Integration](#) chapter.

Middleware (BPEL) Errors

The BPEL processes that make up the middleware layer are the link between the portal application and the back-end system. BPEL processes can execute **Synchronous** or **Asynchronous** requests.

In Synchronous requests, the caller (typically the portal application) is waiting for a response. In Asynchronous requests no process is waiting. Asynchronous requests are typically requests that are stored on a system queue and are picked up by a system process for execution.

The following errors can occur at this layer:

- Errors while attempting to communication with the target back-end system. This is typically a result of an error in the back-end system or wrong configuration of the connection details to the back-end system.
- Back-End system response is a SOAP fault. In this case, the BPEL process will return a special error message to the portal application to indicate that an error has occurred.
- A BPEL process error occurred while executing the web service call received from the portal application. When internal processing errors are encountered, they are caught and a special error message is returned to the portal application.

Important: When errors occur during a **Synchronous** request execution, a special message is returned to the caller (typically, the portal application). When errors occur during an **Asynchronous** request execution, the request is transferred to an Error Queue and an email message can be sent (if configured) to indicate that an error was detected.

Note: For more information about the special messages returned to the portal application by BPEL in case of errors, refer to the [SOA/BPEL Integration](#) chapter.

Portal Application Errors

The portal application includes functionality for the admin user and functionality for the taxpayer using the portal services. The following errors are possible for the portal application:

- **Validation errors.** These are standard local errors related to system settings or configuration maintenance. These errors also include business rules embedded into the admin maintenance logic (for example, duplicate key checking, field value interdependencies, etc). System messages will be displayed to guide the user on how to correct the error.
- **Validation Rules errors.** These are errors that are triggered as a result of a **Validation Rule** associated with a **Field** in the system. The message associated with the rule will be displayed. For more information about Validation Rules, refer to the *Settings and Configurations* chapter.
- **Errors while trying to communicate** with the middleware layer (BPEL) when executing a web services call. In this case a special message will be displayed to indicate that an error has occurred. The exact nature of the error is not communicated in this case and additional investigation is usually required. Typical causes for such an error are:
 - Connection to middleware is not configured correctly.
 - Middleware is not responding or is down.
 - BPEL process returns a SOAP fault due to a system crash.
- Response received back from middleware (BPEL) as a result of a web service call is invalid. In this case the XML response document that is received after a web service call does not contain a confirmation or an error message. In this case a special message will be displayed indicating that there were issues detected with this request. In this case additional investigation is required to ascertain the cause.

Application Messages Overview

Base product application messages communicate information about the requested action. They can contain confirmation information, actual data or error messages.

Application messages are created in multiple places in the system:

- The **Revenue Management Back-End System** - A result of a web service request sent as part of the service supported by the base product (e.g., a payment request).
- **Middleware (BPEL)** - A translation of a message originated in the revenue management back-end system or a message created in this layer due to processing exceptions.
- **Portal Application** - Provides an immediate response to an action performed by the user.

Revenue Management Back-End System Messages

The back-end system responds to web service requests sent by the portal application. The web services that are used by the base product include the structure for the information returned to the caller.

The information that is returned can include many data elements but it will always include messages to communicate the result of the service call.

There are two types of messages returned by the back-end system:

- Confirmation Messages communicate the result of the request. Confirmation messages have the following structure:
 - Confirmation header message
 - Zero or more confirmation details messages
- Error Messages indicate the failure reason if the request cannot be executed.

Important: Confirmation messages don't have to always be "positive" in nature. There could be case that a certain request was processed in the back-end system but the result of the process is rejection of the request or even an error. In this case, the confirmation is that the request was accepted in the back-end system and was processed.

The base product web services definitions (WSDLs) support one error message and multiple confirmation messages (a header message plus a set of details messages). If an error message is returned, the base product portal application does not expect confirmation messages as well, and vice versa.

Back-End System Messages include:

- **Message Category** – A number or a string depending on the revenue management back-end system. The message category is NOT translated or considered in the portal application, it is for information purposes only.

The message category is considered in the translation process (in the SOA/BPEL integration layer) of back-end system Message Numbers to portal message numbers but has no other meaning beyond that.

- **Message Number** – A number or a string.
- Message Parameters:
 - **Parameter Sequence** – A number.
 - **Parameter Type** – The type of parameter being using, for example (Text, Date, etc).
 - **Parameter Value** – Depends on the parameter type.

Middleware (BPEL) Messages

The base product middleware layer is a set of BPEL processes that typically receive the request from the portal application, send it to the target system and communicate the response back to the portal application, if an immediate response is required.

The following scenarios are possible:

- The web service call from the portal application successfully reached the target system and was processed. In this case a confirmation or an error message will be returned to the portal application by the target system (revenue management back-end system).
- The target system cannot be reached or an error occurred while trying to call the target system. In this case a predefined error message will be returned to the portal application.
- The web service call from the portal application doesn't need to wait for a response. In this case a special predefined message is returned by the BPEL process to indicate that the request was received successfully.

According to the scenarios above, it is clear that the base product BPEL processes can also create messages and not just relay the messages that they receive from the back-end system to the portal application.

Note: For more information on BPEL processes logic and messages used, refer to the [SOA/BPEL Integration](#) chapter.

Message Translations and Consolidation

In some case, implementers may choose not to relay the message coming back from the revenue management back-end system directly to the taxpayer (using the portal application). The revenue management back-end system is typically designed for a revenue management system user. Such a user may have access or interest in types of information that should not be exposed to the taxpayer using the portal application.

This is why there is an optional translation process that is implemented by the base product BPEL processes. This process uses DVM tables to translate the message numbers returned by the back-end system to a different set of message numbers to be displayed on the portal application.

The BPEL message translation process enables implementers to:

- Use different message text on the portal application when the verbiage of the back-end system messages is not appropriate for a taxpayer using the portal application.
- Group back-end system messages into one message in order to have a more concise message delivered to the taxpayer.
- Group back-end system messages from several systems into one message. This option is relevant if web service requests can be sent to multiple systems. In that case, messages from different systems may need to be displayed the same way on the portal application. The current translation architecture supports that option.

The base product also allows the back-end system to communicate a message that will be displayed as is without any translations. This is a feature implementers can use if the message must be communicated to the taxpayer as is.

Message translation is done according to BPEL DVM mappings. If no mappings are defined, the messages will not be translated and will be passed to the portal application as they were received from the back-end system. If DVM mappings were provided but the message from the back-end system has no mapping defined for it, a default substitute message will be returned to the portal application so that a valid message is displayed.

Note: For more information about DVM and message translations in BPEL processes, refer to the [SOA/BPEL Integration](#) chapter.

Portal Application Messages

The portal application has the following types of messages:

- Messages displayed as a result of user input. This can be during admin data maintenance (system settings or configurations) or during functional transaction interactions (e.g. input for a refund status request).
- Messages displayed as a result of a web service call that was executed during a transactional request (e.g. make a payment or inquire about refund status). These types of messages can have message parameters that should be displayed with the message.
- System error messages as a result of a failure to perform or complete an operation, for example:
 - If the middleware processes cannot be reached, web service calls will fail.
 - If some system options are missing or not set up correctly, some operations may fail.

Displaying Web Services Messages

When a message is returned from a web service call, the following logic is used:

- If the message number received does not exist in the portal application message definitions, a special system messages is displayed instead.
- Duplicate messages as part of the confirmation details messages will be suppressed. A message will be considered duplication *only* if the message number and parameters are identical to other messages.
- When a message is displayed the message text (defined in the Message entity) is used and the parameter values (returned as part of the web service call response) replace the substitution variables notation **{x}** in the defined text. The substitution is done according to the parameter order. Substitution variables with no value will be replaced by **blanks**. Parameter values with no corresponding substitution variables will be ignored.

Portal Messages Caching

Messages on the application portal are implemented using the Java Resource Bundle. This means that changes made to the **Message** admin data on the portal application are not immediately reflected on the portal. In order for these changes to be reflected the portal application has to be restarted.

Portal Application

Overview

The base product portal application is built from a set of Functional Pages and Admin Pages. The portal application also includes a special **Login Page** for admin users.

All base product pages have the same base product page template and therefore share a common structure.

The Portal Page Template

The portal page template includes the following four sections:

The first section of the page template includes:

- The top bar image and company logo.
- Current User ID information, a link to the WebCenter Administration Console (for customizations) and language selection links.

The second section is a dynamic navigation bar that can include links to base product pages and custom pages. The values displayed will depend on security settings.

The third section follows after the main navigation bar and is used for the page content, which varies for each specific page.

The fourth section includes:

- A custom icons bar that can be configured to show icons to common websites, such as Facebook.
- A custom links bar that can be configured to show links to common pages or websites that provide general information about your organization. Examples of such links include contact details, privacy statements, etc.
- A copyright message that appears on all pages. This message has a default value that can be modified.

Portal Admin Pages

The base product admin pages are divided into two categories: **Settings** and **Configurations**.

The settings pages define general system options that are used across the portal application.

The **Configurations** pages define more specific options for particular services that are supported by the base product.

When selecting either of the above options from the main navigation bar an "entry" page is displayed that has the following structure:

- The top and bottom sections of the page are taken from the page template.
- The left side contains the specific navigation links which list all the possible pages under this category.
- The middle of the page is a place holder for customer specific HTML content.

When a user selects one of the links on the left side of the page the middle part of the page (the place holder) is replaced with specific functionality related to the option selected. For example if the user selects **System Options**, the middle part of the screen switches to include the system options search and maintenance page.

Portal Functional Pages

The base product functional pages include the **Home** page, the **Online Services** page, and the specific functional pages for various supported services.

The **Home** page and **Online Services** page have a similar structure to the admin "entry" pages:

- The top and bottom sections of the page are taken from the page template.
- The left side contains the specific navigation links area which lists all the possible pages available from this page.
- The middle of the page is a place holder for customer specific HTML content.

The functional pages have the same structure as the admin specific pages when a page is selected from the left side navigation links area. In this case, as with admin pages, the place holder for custom content in the middle of the page is replaced by the specific base product provided functionality (for example, service requests, payments, etc).

Customizing the Portal Structure and Pages

Implementers can customize the portal structure and define the content to be displayed in all the pages that have place holders designed for custom content. In addition the layout of the template page and other pages can be modified and additional content can be added to all or some of the pages. New pages can be created and they can have links to the base product pages.

For more information about the customization options of the portal application, see the chapter, "Customizing the Self Service Portal Application".

Monitoring the Application

The self-service application involves different and distributed systems, and the root cause of issues is sometimes difficult to identify. Monitoring of key elements can help isolate issues and make them easier to address.

For additional information, see [Monitoring the Status of Oracle Fusion Middleware](#) in the *Oracle Fusion Middleware* online documentation library.

Monitor the Portal Application Using Enterprise Manager

1. Log in to Oracle Enterprise Manager as a system admin user.
2. From the **Domain** menu (e.g., Farm_<<domain_name>>) on the left side of the screen, expand **Application Deployments**.
3. Click on the portal application entry (e.g., "WSSPortal_application...") to load the portal application summary page.
4. Monitor the **Response** and **Load** graphs to get an overall idea of how the application is performing.
5. To drill down further into the log information, click **Performance Summary** from the **Application Deployment** menu at the top of the screen.

This loads another page with more statistics and other graphs to active **Sessions**, **Request Processing Time**, **Request (per min)**, etc.

6. To monitor how each page in the portal application is performing, click the **Application Deployment** menu at the top of the screen and select **WebCenter Portal > Page Metrics**

This action opens a page that lists the processing time required to load each portal page.

Monitor Using WebLogic Logs

WebLogic logs can be monitored to get more information on exceptions and application status.

Logs can be monitored using either Oracle Enterprise Manager or by directly accessing the physical machine on which the managed servers are running. Logs monitored from Oracle Enterprise Manager are more interactive and allow search capabilities, making it easier to diagnose an issue quickly.

Command-line administrators can also directly use the logs on the physical machine.

Monitor Portal Logs Using Enterprise Manager

1. Log in to Oracle Enterprise Manager as a system admin user.
2. From the **Domain** menu (e.g., Farm_<<domain_name>>) on the left side of the screen, expand **Application Deployments**.
3. Click on the portal application entry (e.g., "WSSPortal_application...") to load the portal application summary page.
4. From the **Application Deployment** menu at the top of the screen, select **Logs > View Log Messages** to load the **Log Messages** page.
5. Select the criteria from the form, e.g., set **Date Range** to **5 hours**, then click **Search**.
6. Select any row in the table showing all log entries to load the details in the bottom preview pane.

Tip: Click the log file name to get additional log information for your selection.

Monitoring Document References

Consult the following documentation for information on monitoring document references [Monitor Oracle Fusion Middleware](http://download.oracle.com/docs/cd/E17904_01/core.1111/e10105/monitor.htm#CFAEHCGG) (http://download.oracle.com/docs/cd/E17904_01/core.1111/e10105/monitor.htm#CFAEHCGG).

Chapter 3

Settings and Configurations

System Administration Pages Overview

The portal application administration pages (commonly referred to as "admin pages") allow the user to configure the system to support the services that are included in the base product, as well as to create new services.

Admin pages are divided into two groups:

- The **Settings** group includes general setup information that supports the portal application and is not specific to any of the services.
- The **Configurations** group includes a specific setup for the business functionality provided in the application. The configurations are explained in details in each corresponding functional services chapters.

All admin pages that support configuration management have a common pattern that is described in this section.

When configuring a system or a service option the following pages are typically available:

- **Search Page.** This is the first page that the user encounters. In this page, the user can search for existing configurations and select the action required.
- **Maintenance Page.** This is the page where configuration details are defined. This can be a single- or multi-tab page, depending on the complexity of the entity being configured.

Admin Search Page

The search page is the first page displayed when configuring a system option. This page allows the user to search for existing definitions.

The search page has a search section containing the fields for the search criteria and a result list area where the search results are displayed.

Common Actions

Search – Executes the search according to the specified search criteria.

Reset – Resets the search criteria to the default values.

Sorting – Columns that appear in the result list are sorted by clicking on either the ascending or descending sort button on the column header. The result list can only be sorted one column at a time.

Selecting a Record – Clicking on any part of the row selects the record. The selected row is highlighted in grey.

Note: When hovering on a row, the row is highlighted in light blue.

Add (the + button) – Adds a new record.

Edit (the pencil button) – Edit the selected record. A record can be also edited by pressing the value of the first column, if displayed as a hyperlink.

Delete (the X button) – Deletes the selected record.

Owner Flag

Some admin entities have an additional field that records the owner of that admin record. The owner flag value is either **Oracle Tax Self Service** (also referred to as **Base Product**) or **Customer Modification**. Customer-owned records are typically defined by the user or implementer and allow all the common actions. Base product records allow a limited set of actions:

- **Add** is always allowed. When adding a new record the owner flag on the record is assigned the value of **Customer Modification**.
- **Edit** may be allowed for base product records in some cases. Edits are usually allowed only on certain details, such as override descriptions.
- **Delete** is never allowed for base product records. When a base product record is selected, the delete button is hidden.

Admin Maintenance Page

The maintenance page allows the user to enter or change the details of a new or existing record.

Admin maintenance can be a **popup window** or a **full page**.

When editing an existing record's details, if the record is owned by the base product, some of the details displayed on the page may be read-only.

Full page maintenance pages can have a single tab or multiple tabs.

Maintenance pages can also include a list of dependent values. The list of dependent values is displayed as a search result and usually supports adding, editing, or deleting a value, depending on the owner of the record and the dependent value.

Popup Window Actions

OK – executes records validations, and, if no errors are detected, saves the record and closes the popup window.

Cancel – cancels the operation and closes the popup window.

Full Page Actions

Save – Saves the changes made by the user.

Reset – Restores the value of the entity being maintained to their state before changes were made.

Search – Returns the user to the search page.

Search Dialogs

The primary identifiers of the admin entities are displayed in maintenance pages with a **magnifying glass icon** next to them. This icon initiates the **Search** action.

The **Search** action opens a popup window with search criteria fields and a result list. When a value is selected from the result list it is returned to the field that initiated the search.

Defining System Options

System options setup can be reached from the **Settings** page. It defines a number of global configuration values used by the portal application at run-time.

Several options are provided with the base product, and the default values can be adjusted to fit your organization needs:

- **DEF_DATE_FORMAT** defines the date format that the portal application will expect when entering date values. The valid values for this option are: **DMY** (for a DD/MM/YYYY format, **MDY** (for MM/DD/YYYY format), **YMD** for (YYYY-MM-DD format). Additional values can be configured using the **DATEFORMAT Lookup**.
 - **Please note that the date format is defined in the *Extended Value* of that lookup definition. The extended value is the actual format that is used by the system.**
- **DEF_BACKEND_DATE_FORMAT** defines the date format that the portal application will use when communicating with the revenue management system. The valid values for this option are the same as in the **DEF_DATE_FORMAT** option.
- **DEF_LOCALE** defines the default Language to be used in the portal application. The valid values for this option are **Language** code values.
- **DEF_CURRENCY** defines the default currency that is used in the portal application for fields displaying money amounts. The valid values for this option are defined in the **CURRENCY Lookup**. The default value delivered is **USD**, additional values can be defined as needed.

The following options should be configured in the system to support sending mail from the portal application:

- **MAIL_SMTP_HOST** defines the SMTP server host name. This option required freeform text value and no validation is performed on that value.
- **MAIL_SMTP_PORT** defines the port number of the SMTP server. This option required freeform text value and no validation is performed on that value.

System Options List

The user can review existing system options.

The standard **Edit** and **Delete** functions are available for each system option.

You can click the **Add** button in order to define a new system option.

System Options Maintenance

Configuration Option is the type of system option that is being defined. The valid values for this option are defined in the **CFGOPTION Lookup**.

The **Option Value Type** defines the type of values that are expected for the system option. The valid values for this field are **Freeform Text** or **Lookup**. There is no specific validation of the values entered beyond the value type validation.

The **Freeform Value** field will hold the system option value if **Freeform Text** was selected as the option value type.

The **Lookup** field will hold the lookup code value if **Lookup** was selected as the option value type.

The **Lookup Value** field will hold one of the possible values of the selected lookup code, if **Lookup** was selected as the option value type.

Important: You should **NOT** change the option value type for provided system options with option value type of **Lookup**.

Defining Languages

The language setup can be reached from the **Settings** page. It allows the user to define the languages that are supported on the portal application.

Note: Adding a language in this page is the first step in enabling a new supported language. See [Supporting Additional Languages](#) for more information on the steps involved in adding a new supported language.

Language Search

The user can search existing language definitions.

The standard **Edit** and **Delete** functions are available for each language definition.

Click the **Add** button to define a new language.

Language Maintenance

Locale defines the ISO language code that will appear on the upper right part of the portal pages for switching between languages.

Important: You should not define country-specific language codes. For example, the code "en" is for English and should be defined once. You should *not* use "en_US" or "en_CA" for USA or Canadian English.

Reading Direction is not currently in use.

The **Owner** indicates whether the language definition is owned by the base product or by the implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when implementer defines a new language. This information is display-only.

The **Supported** checkbox indicates whether this language can be used (e.g., it can be switched to by the user or by the taxpayer).

Defining Messages

The portal application communicates informational or errors to the end user via messages. The base product is provided with a set of messages that can be modified by implementers. In addition user defined messages can be added to the system.

Important: Since messages are cached in the system, changes in message definitions require an Application Web Server restart for the changes to take effect.

Messages that are defined in the portal application are being using in the following way:

- Message numbers may be returned as part of a web service response. The corresponding message text is retrieved by the portal application and displayed to the taxpayer. Implementers can override a provided message text.

- When internal error occurs in the portal application, information is conveyed to the user using messages. The messages are defined and owned by the base product, but the actual message text displayed to the user can be modified.
- Messages can be associated with **Validation Rules**. When defining a validation rule, implementers can use one of the messages provided in the base product or create their own message.

Note: For more information about message translations and revenue management system web services, see [Message Translations and Consolidation](#).

The message setup can be reached from the **Settings** page.

Message Search

The user can search existing message definitions.

The standard **Edit** and **Delete** functions are available for each message definition.

You can click the **Add** button to define a new message.

Message Maintenance

Message Number is the identifier of the message.

Important: The base product is provided with messages in the range of 0-89,000. Customer-defined messages should use the number 90,000 and above.

The Message Category defined the type of message. It is currently used for information purposes only. Valid values for the category are: **Information Messages**, **Warning Messages**, **Error Messages**, and **System Fatal Error**.

The **Owner** indicates whether the message definition is owned by the base product or by the implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when the implementer defines a new message. This information is display only.

The **Message Text** is where you define the text that will be displayed to the user. You can use the {n} notation within the message text to cause message parameter values to be substituted into a message.

Example: The message text, "The payment of {1} received on {2} will be posted on your account " has the values of two parameters merged into it before it is displayed to the user (%1 is the payment's amount, %2 is the date of the payment).

The values are translated based on the parameter type supplied together with the parameter value. Supported parameter value types are defined in the lookup **MSGPARMTYPE**.

Note: The system merges whatever values are supplied to it. Therefore, if a system (or a web service response) supplies an amount as the second merge parameter in the above message, this amount is merged into the message at the second place (rather than the date).

The **Override Message Text** is applicable if you are modifying a base product message definition in order to change the text displayed to the portal page.

Defining Lookups

A lookup is a set of discrete values grouped together under a common name: the **Lookup Code**.

Lookups are used in the portal application to allow a user or a taxpayer to choose from a list of possible values in certain fields. Special lookups are documented within functional section they belong to.

The base product is provided with a number of a pre-defined lookups. Some of these lookups can be modified by implementers to include additional values. In addition, new lookups can be defined and used when defining **Fields**.

The lookup setup can be reached from the **Settings** page.

Lookup Search

The user can search existing lookup definitions.

The standard **Edit** and **Delete** functions are available for each lookup definition.

You can click the **Add** button to define a new lookup.

Lookup Maintenance

Lookup Code is the name for the lookup.

The **Owner** indicates whether the message definition is owned by the base product or by the implementation (**Customer Modification**). The system sets the owner to **Customer Modification** when the implementer defines a new message. This information is display-only.

The **Override Description** field is only applicable when modifying a base product lookup.

The **Customizable** checkbox indicates whether new values can be added to this lookup. This is mostly applicable for base product lookups that allow customers to add additional values on top of the base product provided values.

Help text can be added to any lookup and while it is not displayed on the base product portal pages, it can be used in custom developed pages. The help text can include HTML.

The **Override Help** field is only applicable when modifying a base product lookup.

Lookup Value List

For each lookup the user can view the defined lookup values associated with the lookup code.

The standard **Edit** and **Delete** functions are available for each lookup definition.

You can click the **Add** button to define a new lookup value.

Lookup Value Maintenance

Lookup Value is the value that will be used when the entry is selected.

The **Active** checkbox indicates whether this value entry will be visible in the portal application.

The **Description** is the value displayed for this entry on the page (in the selection list).

The **Override Description** field is only applicable when modifying a base product lookup value.

The **Image URL** is currently not in use.

Extended Value is additional information provided for the lookup value. This value is used by the system in special cases, for example, lookup code **DATEFORMAT** (see *Defining System Options*).

Help text can be added to any lookup value. The help text can include HTML. This help is currently only displayed on the base product portal pages in special cases.

Example: Help for values of lookup code **TENDERTYPE** is displayed on the portal page when the taxpayer selects the type of payment to make.

The **Override Help** field is only applicable when modifying a base product lookup value.

Important: After adding or modifying a lookup value, the **Save** action (on the lookup value maintenance screen) should be used to save changes.

Note: The **Save and Add New** action is only applicable for lookups. It is *not* applicable for lookup values.

Defining Fields

Fields are used to define the input elements placed on the dynamic UI pages, such as taxpayer service request. Field configuration controls the input element's appearance on the screen and the input value format.

Note: Fields are reusable. Some Fields can be referenced in multiple service request definitions or even included multiple times within the same service request definition.

The base product is provided with a number of fields and new fields can be created by implementers.

The field setup can be reached from the **Settings** page.

Field Search

The user can search existing field definitions.

The standard **Edit** and **Delete** functions are available for each field definition.

You can click the **Add** button to define a new field.

Field Maintenance

Field Name is the identifier of the field.

Description is the label displayed next to the input field on the dynamic screen Data Type defines the input type for that field. The data type controls how the field is displayed on the page and what types of input are permissible. Valid values for data type are:

- **Boolean** – Displayed as a checkbox.
- **Number** – Displayed as a number.
- **Currency** – Displayed as a numeric amount.
- **Date** – Displayed as a date according to the format defined in the portal application system options (**DEF_DATE_FORMAT**).
- **Lookup** – Displayed as dropdown list according to the values defined for the lookup code (that is defined for this field).
- **Text** – Displayed as a text field.

The implementer can associate a **Validation Rule** for a field. This validation rule will be executed by the portal application when value is entered in this field. See [Defining Validation Rules](#) for more information.

Display Length is applicable for all data types except for **Boolean** and **Date**. It defines the space reserved on the page for displaying the value of this field.

Field Length is applicable for all data types except for **Boolean** and **Date**. It defines the number of characters that can be entered for this field.

Precision is applicable only for **Number** and **Currency** data types. It defines the number of decimal positions reserved for this field (out of the total field length).

Example: if the field length is 6 and the precision is 2, an example for a valid number will be: "123.45" (since total length includes the decimal point).

Note:

The display length, field length and precision values are not mandatory. If not provided the portal application will apply the following defaults:

Display Length = 80 characters

Field Length = unlimited

Precision = unlimited

The **Hint** defined for a field is a text that will appear on the right side of the input field on the page. It can be used to provide the format that is expected in this field, for example "(999) 999-9999" for a field that expects a telephone number in it.

The **Detailed Description** is for documentation purposes only.

The **Help** text defined for a field is displayed in a popup window when the field is in focus.

Defining Validation Rules

Validation rules are additional conditions evaluated when a value is entered in a field that has an attached validation rule.

Note: The portal application currently supports only one type of validation rule: **Regular Expression**.

The validation rule setup can be reached from the **Settings** page.

Validation Rule Maintenance

The user can search existing validation rule definitions.

The standard **Edit** and **Delete** functions are available for each validation rule definition.

You can click the **Add** button to define a new validation rule.

Validation Rule Maintenance

Validation Rule is the identifier of the rule.

The **Validation Expression** has to be written using a standard **Regular Expression** syntax.

Example: "[A-Z] [0-9]" will verify that the input is a single letter followed by a single digit.

A **Message Number** is associated with the rule to indicate what message to display when the input value fails the validation

Chapter 4

Service Requests

Overview

Taxpayers may have many reasons to contact a revenue authority, such as to communicate changes in personal information, to request a tax certificate, to raise a dispute over calculated assessments, charges or other financial details, etc.

The Service Request feature is designed to implement this type of communication using configurable elements. The feature enables users to create a template for each type of request that governs a number of common attributes and steps, such as:

- Defining the fields that capture required data for the request
- Defining the user interface for data entry
- Defining field level validation rules for the data
- Specifying other business rules such as whether the taxpayer identity must be verified for this type of request, whether the request should be directed to specific revenue authority recipients or to a backend application for automated processing, etc.

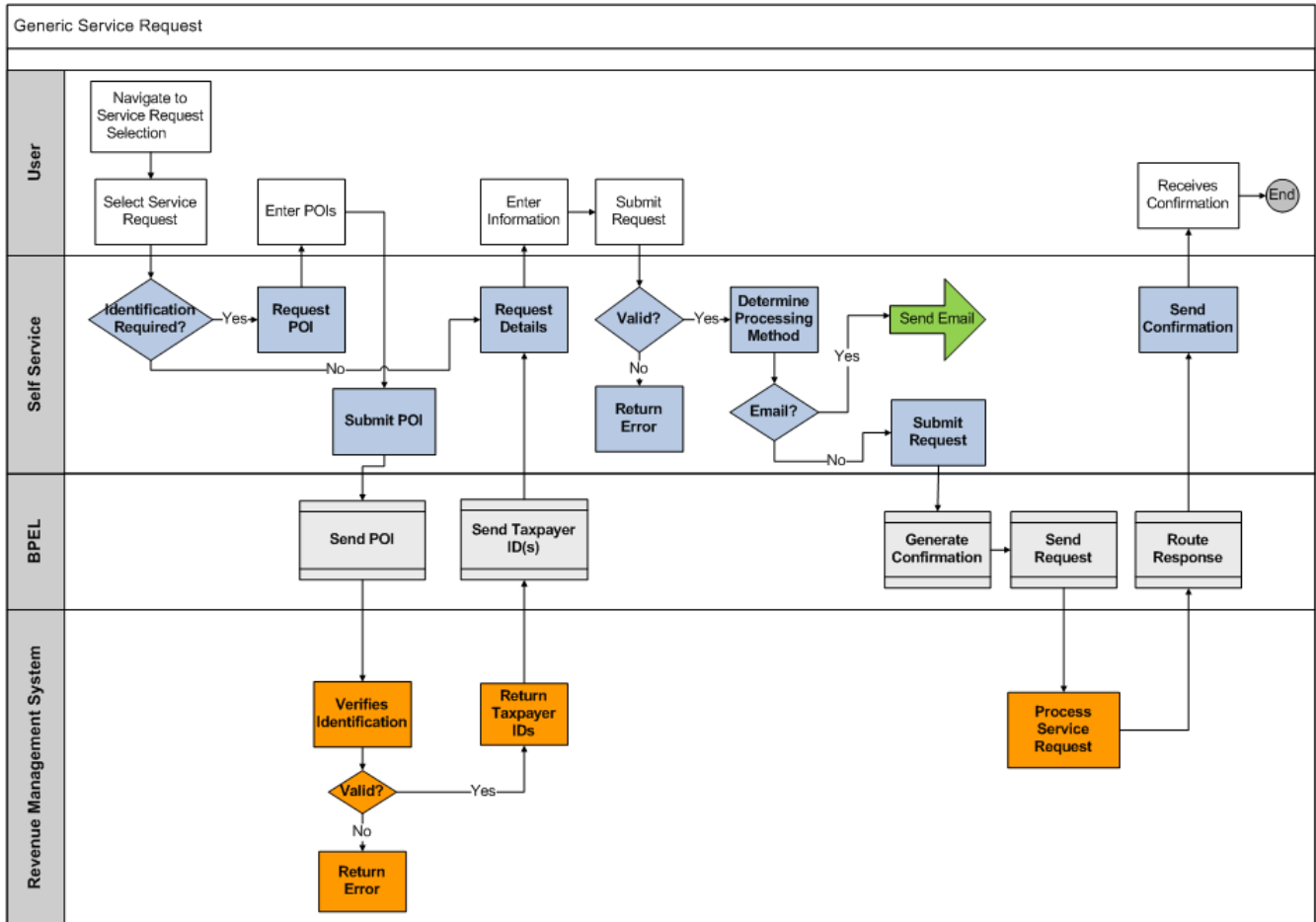
Service Requests can be grouped into categories. This enables taxpayers to find their particular request more quickly by first choosing an appropriate category and then selecting a request within that group.

Service requests can be accessed in one of two ways:

- Via the **Service Central** option. This option is accessible from the On-Line Services pull-down menu on the web self service portal navigation bar and from the left-side navigation panel.
- Via direct hyperlinks. Links to a specific request, such as “mail us your suggestion”, “request payment plan”, or “check eligibility to participate in a program” can be enabled from any page in the portal application.

Generic Service Requests

Process Flow



Service Request Selection

On the initial Service Central screen, the taxpayer is asked to select a service category first and then to choose a specific topic.

The categories and topics appear on the screen as a dropdowns. Each topic corresponds to an active Service Request.

After a topic is selected the system reads the Service Request configuration. It first determines whether this service request requires taxpayer identification. If so, the system reads the Identification Request definitions and invokes a Taxpayer Identification sub-flow. If the taxpayer identification step was successful, the request header is populated with taxpayer IDs. If identification is not required, the system continues to the Main Request Input step.

Main Request Input

The main service request screen is rendered dynamically according to the service request configuration. The taxpayer is prompted to enter the information required for this specific service.

Note:

On request submission a web service (**TSTaxpayerServiceRequest**) is triggered. The request contains a collection of sequence/field name/field value entries.

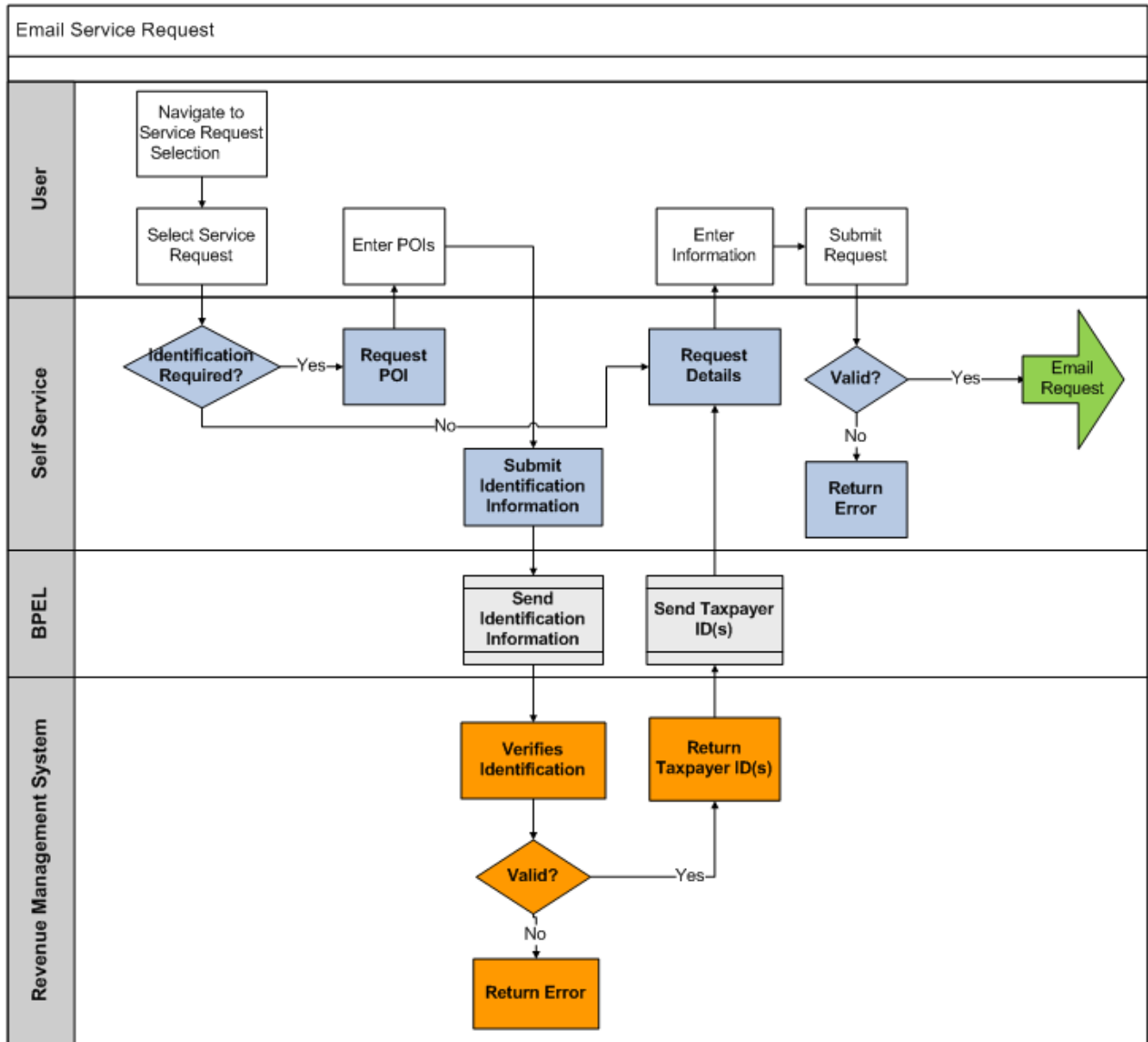
If the web service call was successful, the response contains the confirmation.

Confirmation

The expected web service response contains Confirmation number and message(s). This information is displayed to the taxpayer. Confirmation may be printed for the taxpayer's reference.

Emailing a Service Request

Process Flow



Some Service Request types are configured to be delivered via email. No web service interaction is involved. The first two steps are similar to the Generic Service Request.

Service Request Selection

On the initial Service Central screen taxpayer is asked to select a service category first and then to choose a specific topic. The categories and topics appear on the screen as a dropdowns. Each topic corresponds to an active Service Request.

After a topic is selected, the system reads the Service Request configuration. It first determines whether this service request requires taxpayer identification. If so, the system reads the Identification Request definitions and invokes a Taxpayer Identification sub-flow. If the taxpayer identification step was successful, the request header is populated with taxpayer IDs. If taxpayer identification is not required, the system continues to the Main Request Input step.

Main Request Input

The main service request screen is rendered dynamically according the service request configuration. The taxpayer is prompted to enter the information required for this specific service.

Note: The taxpayer's e-mail address should be a required part of the service request input. See the [Defining Service Request](#), (process method configuration) for more details.

Email the Request

The target (“From”) email address is taken from taxpayer’s input. The recipient (“To”) email is derived from the service request configuration. The SMTP server details are derived from the System Configuration Options. The service request information is e-mailed to the recipient.

The message body is generated from the list of the fields defined in he service request. The message body is formed by appending field names and their associated values for each field. If the field type is a lookup, the description is retrieved for the selected lookup value. The email body is emitted to the responsible email handler class along with details such as recipient, subject of the email, etc.

The email handler is responsible for retrieving details such as the SMTP host and port which are determined from system configuration options.

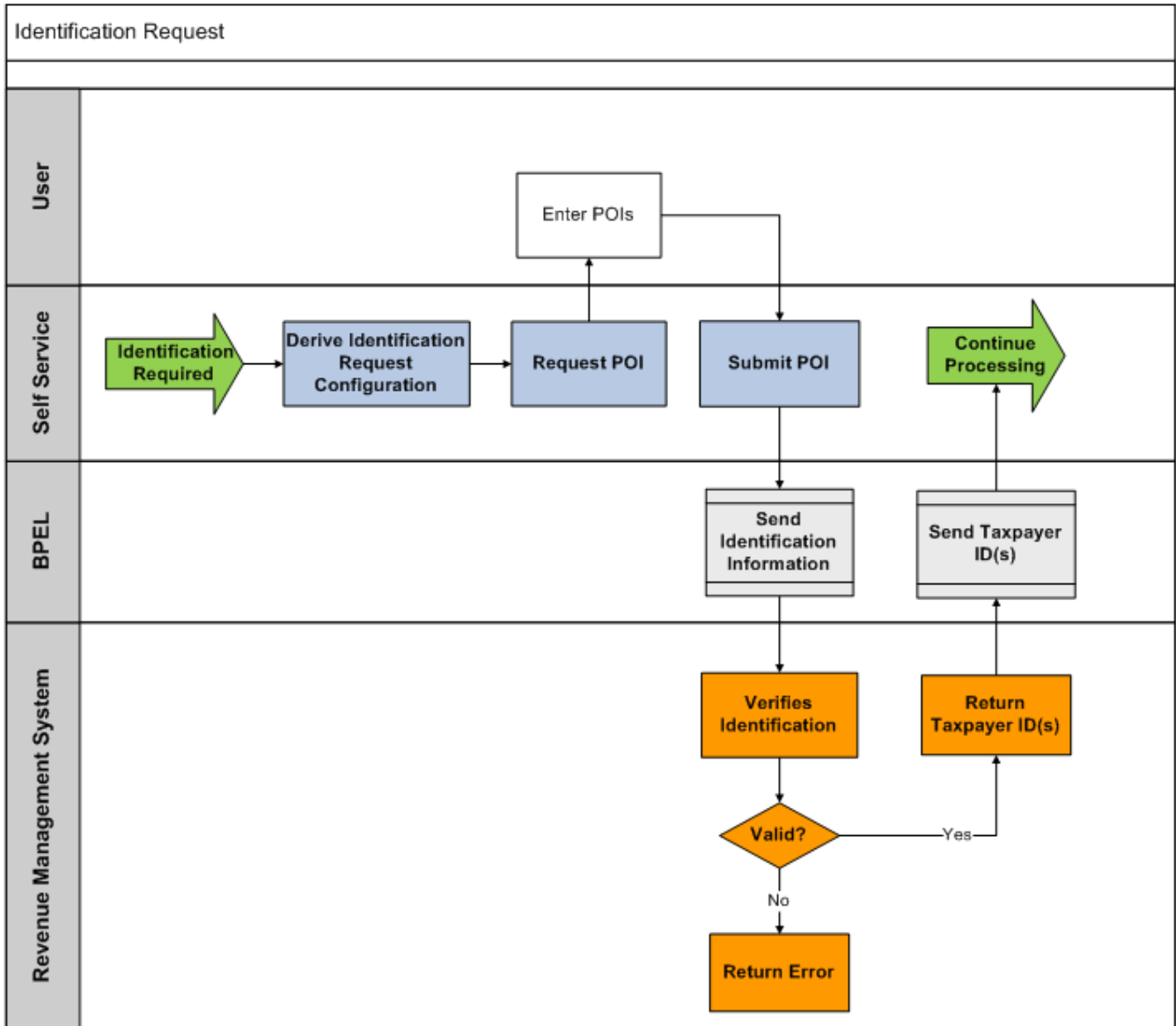
Emailing the Request

The message body is generated from the list of the fields defined in service request. Message body is formed by appending field name and its value for each field. If field type is lookup, description is calculated for the selected value. Email body is emitted to the responsible email handler class along with the details like recipient, subject of the email etc.

Email handler itself is responsible for calculating details like SMTP host and port which are determined from system configuration options details.

Taxpayer Identification Request

Process Flow



Initiate Taxpayer Identification

A Taxpayer Identification request is initiated from other components as a sub-flow. The system reads the identification service request definitions and the input Taxpayer Identification screen is rendered dynamically.

Taxpayer Identification

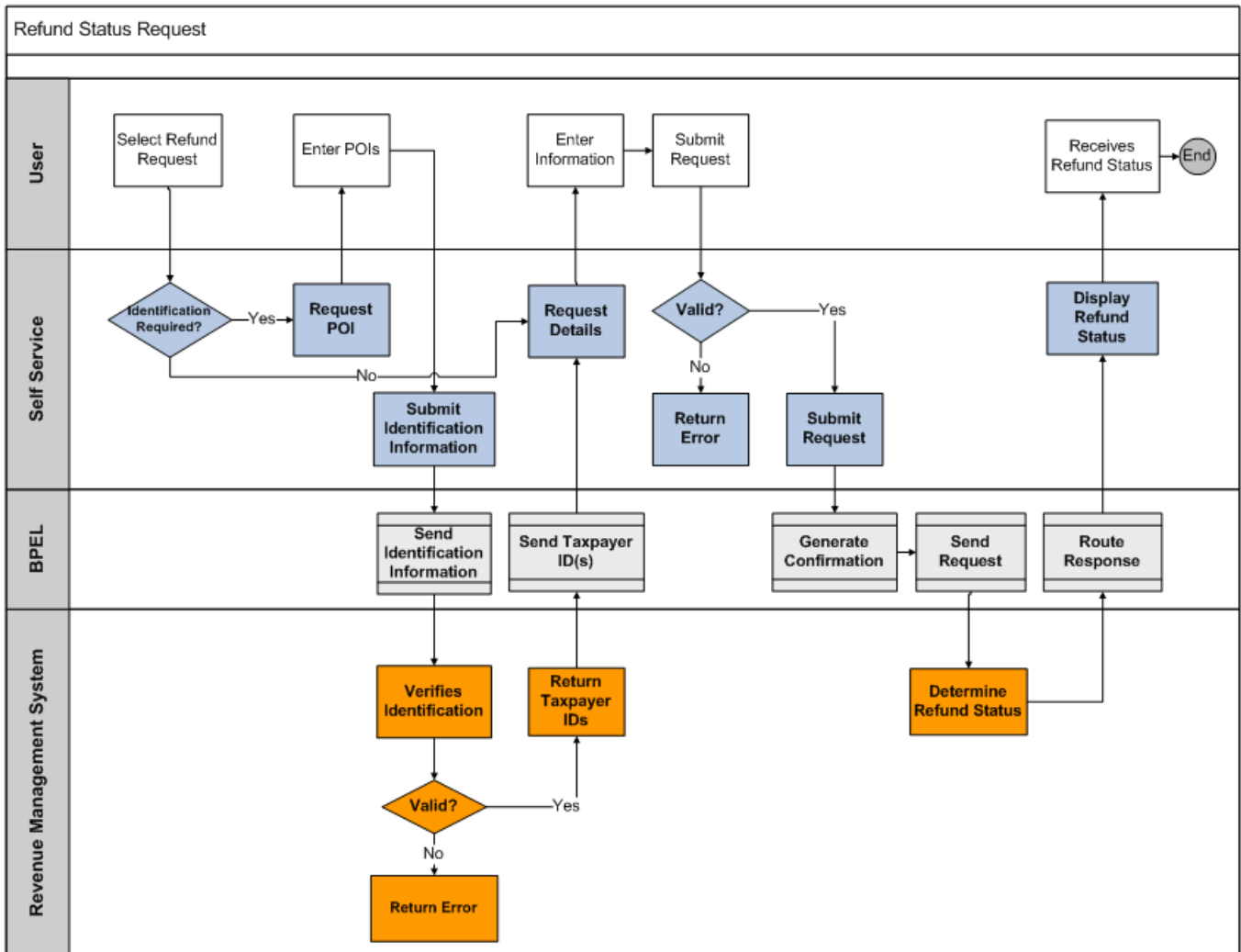
The taxpayer is asked to provide Proof of Identity (aka POI) details. The taxpayer enters the identification information and submits the request.

Note: On request submission a web service (**TSTaxpayerIdentification**) is triggered and the collected input is sent to the back-end system for verification. The successful response contains a collection of taxpayer IDs.

Refund Status Inquiry Request

Process Flow

Refund Inquiry requests belong to a special category of requests and get invoked from the dedicated online page.



Refund Request Selection

On the initial screen, the description of each active service request of the special category Refund Status is displayed as a hyperlink. The taxpayer initiates the refund status inquiry by clicking on the associated link.

When the taxpayer clicks the hyperlink, the system reads the Service Request configuration. It determines whether this refund request requires taxpayer identification. If so, the system reads the Identification Request definitions and invokes a Taxpayer Identification sub-flow. If the taxpayer identification step was successful, the request header is populated with taxpayer IDs.

If taxpayer identification is not required, the system continues to the Main Request Input screen.

Main Request Input

The main service request screen is rendered dynamically according the service request configuration. Taxpayer is prompted to enter the information required for this specific service.

Note:

On request submission a web service (**TSGetRefundStatus**) is triggered. The request contains a collection of sequence/field name/field value entries.

If the web service call was successful, the response contains the confirmation and three amounts: refund, carry-over, and offset.

Confirmation

The expected web service response contains Confirmation number and message(s). This information is displayed to the taxpayer. Confirmation may be printed for the taxpayer's reference.

Defining Service Requests

A Service Request configuration defines what input the taxpayer should supply in order to receive the service and how this input should be presented on the UI. It also defines request delivery method and processing flow.

Service Request Search

The user can search for an existing Service Request.

The standard **Edit** and **Delete** functions are available for each Service Request.

You can click the **Add** button to define a new Service Request.

Service Request Maintenance

Service Request: Main

Request Type is a user-defined user-defined code that uniquely defines the service request type.

Active indicates whether this Service Request is ready for use.

Description is text displayed for this entry on the Service Request selection page.

Detailed Description is for internal use.

Processing

Category is used to group logically related or similar requests. It can also represent a specific business feature, such as Refund Status Inquiry or Taxpayer Identification. The categories are defined using lookup SRCAT. Users can add a new category, if needed.

Response Mode controls whether the request will be sent synchronously to the Revenue Management System or placed into the queue (asynchronous mode). This option defines if the taxpayer will receive an instant response or just an acknowledgement from the SOA layer. In the latter case, the request is placed in the queue and later transmitted to the Revenue Management System. The decision should be made based on the anticipated volume of requests of this type and on the nature of the processing in the back-end system. This option is applicable if the **Process Method** is a web service or both email *and* a web service.

Process Method defines where and how to send the request. The request can be routed to the revenue management system via a web service call, or emailed to a specific revenue management support group, or both.

Recipient defines a support group dedicated to handle requests of this type. Use the **RECIPIENT** lookup to maintain a list of support groups and their respective email addresses.

Identification Requirement

The **Identification Required** option indicates whether the taxpayer's identity must be verified prior to providing the service.

Applicable To indicates whether the identification requirements are applicable only for casual (not signed in) website users or should always be performed.

The **Identification Request** is a dropdown list that presents a selection of taxpayer identification service requests. At run-time this identification request is invoked and a taxpayer is prompted to enter Proof Of Identity (POI) information.

Service Request: Appearance

Display Text Area? Indicates whether the request should include a multi-line text field where taxpayer can enter comments or other long text input.

Text Area Label defines the label to appear next to the text area.

Text Area Length defines the maximum length of the text.

Email required? Indicates whether this request should include the prompt for an email address.

Note: The email address entered in this predefined field will be transmitted to the revenue management system under the <head> element.

Email Label defines the label for the email address input field.

Header Text appears on the top of the dynamically rendered request screen, above the request fields. It can be entered as HTML or as plain text.

Footer Text appears at the bottom of the dynamically rendered request screen, below the request fields. It can be entered as HTML or as plain text.

Help appears on the **Payment Details** page beneath the payment destination fields. It can be entered as HTML or as plain text.

Service Request: Fields

The Payment Destination's **Fields** list is displayed on this tab. Fields represent the information that should be collected from the taxpayer in order to process this specific type of payment.

The standard **Edit** and **Delete** functions are available for each field on the list.

You can click the **Add** button in order to define a new payment destination field.

Request Field

Request Sequence uniquely identifies the field in the list. It is copied to the web service request paymentDestinationDetails collection and sent to the back-end system.

Field is referencing a Field entity. By default, the properties of this Field are used to render the input UI.

Display Sequence controls the order of the input fields on the screen.

Mask indicates whether the input value should be masked on the UI.

Required indicates whether the input value is required or optional.

Validate indicates whether the input value needs to be validated.

Request Validation references the Validation Rule attached to the request field. If populated, it overrides the validation rule attached to the Field entity.

Request Description provides an alternative label for the field on the screen.

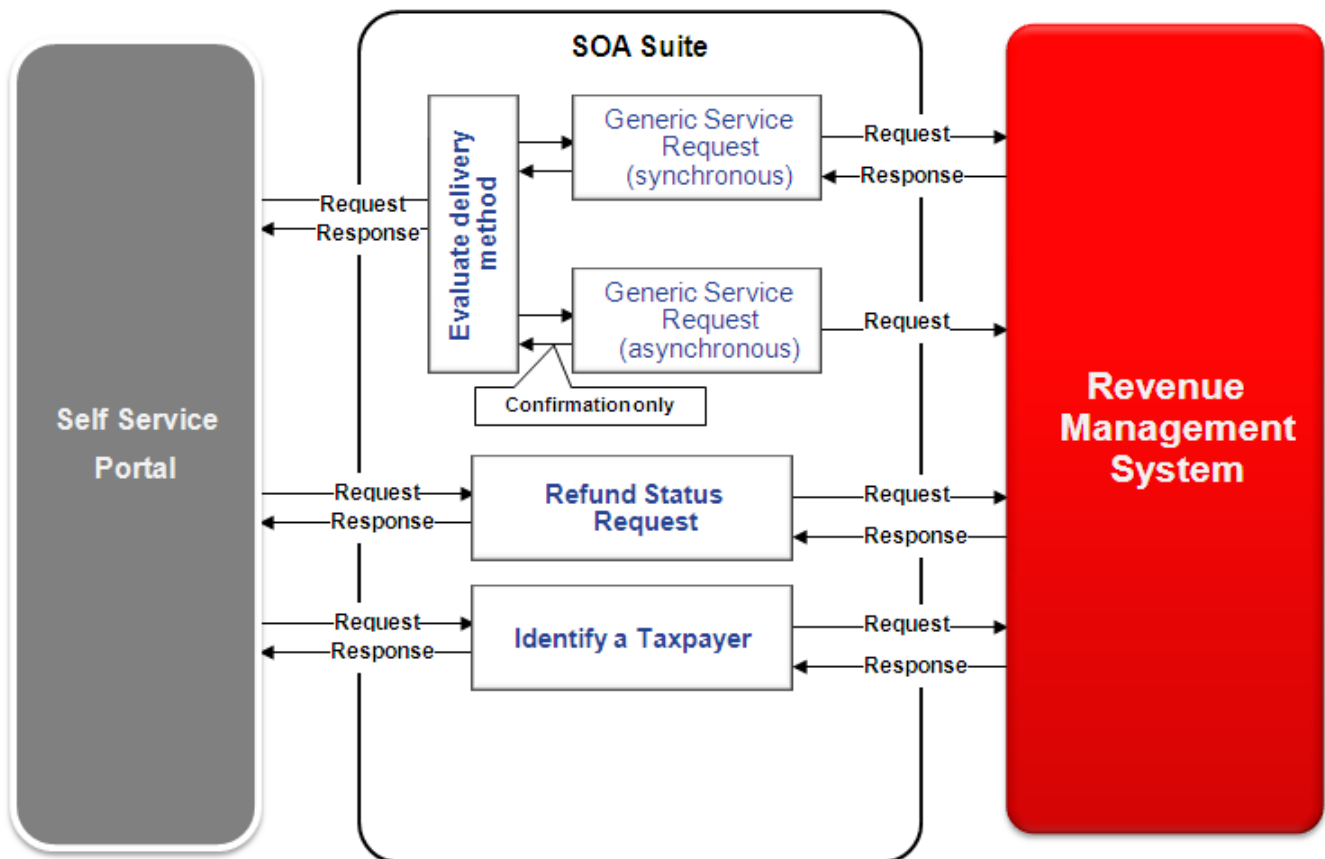
Request Detailed Description provides an alternative description for the internal use.

Request Help provides an alternative popup help text for the field on the screen.

Request Preview

The special Preview action is available on the Service Request maintenance page. Press this button to verify the look of the service request screen. You can fine-tune the screen design, the HTML of the header and footer, check the field's order, etc.

BPEL Processes



The integration SOA composites invoke the revenue management system web services and send the response back to the self service portal application.

The confirmation number is generated when a service request is invoked (with the exception of taxpayer identification requests). This confirmation number is returned as part of the response.

The common features of all provided processes are:

- SOA Composite processes have pre-transformation extension point and post-transformation extension points as explained in section Integration Extensibility.
- SOA Composite processes have extension templates for each level of the outgoing message xml schema as explained in section Integration Extensibility.
- All endpoint invocations are using dynamic partnerlink URL and are be configurable.
- For confirmation number the Confirmation number utility service is called using dynamic partnerlink.

For additional information, see [SOA/BPEL Integration](#).

Implementing Service Requests

This section describes the steps required to implement service request functionality.

Web Services

In order to support taxpayer service requests the revenue management system is expected to implement the following web services:

Web Service	Description
TS taxpayerServiceRequest	<p>Based on the information collected from the taxpayer on the web portal, it should trigger the service and return a confirmation or error with the response. The request message contains a service request type (code) and the collection of sequence/field name/field value details.</p> <p>The request handling in the revenue management system could be as follows:</p> <ul style="list-style-type: none">• Determine the service request type.• Perform the logic associated with this type using request details field values.• Populate the response with confirmation message(s) numbers and parameters.
TS taxpayerIdentification	<p>Validates POI details provided by the taxpayer on the request. The request message contains a service request type (code) and the collection of sequence/field name/field value details.</p> <p>If the taxpayer has been identified successfully, the response should be populated with one or more taxpayer IDs.</p> <p>The POI details evaluation may result in error. If so, the error message number should be populated on the response.</p>
TS GetRefundStatus	<p>The service supposed to evaluate the input provided by the taxpayer and evaluate the status of a potential refund. The request message contains a service request type (code) and the collection of sequence/field name/field value details.</p> <p>The response expected to contain the following:</p> <ul style="list-style-type: none">• Confirmation message(s) describing the refund status.• Additional (optional) information: expected refund receive date and three amounts: refund, carry-over, and offset.

Supported Service Requests

Service Request	Description
TAX_CLEARANCE_CERT – Tax Clearance Certificate Request	<p>Enable user to request and receive a tax clearance certificate.</p> <p>User is expected to enter the following information:</p> <ul style="list-style-type: none">• Email address (Alphanumeric)• Phone number (Alphanumeric)• Tax Certificate Purpose (Lookup)• Paper Copy Required (Boolean) <p>Identification Requirements: IDENTIFY_TAX_CLEARANCE (Identification For Tax Clearance Request)</p> <p>Process synchronously, web service.</p>
REFUND_STATUS_UNR – Check the status of your refund	<p>Allow users to request a status of their refund for a given tax type and tax period.</p> <p>User is expected to enter the following information:</p> <ul style="list-style-type: none">• Taxpayer name (Alphanumeric)• Identifier Type (Lookup)• Identifier Number (Alphanumeric)• Tax Type (Lookup)

Service Request	Description
	<ul style="list-style-type: none"> • Filing Period Start Date (Date) • Filing Period End Date(Date) • Expected Refund (Number) <p>Identification Requirements: none. Process synchronously, web service.</p>
IDENTIFY_TAX_PAYMENT – Taxpayer Identification for Payment	<p>Provides the ability to validate taxpayer POIs required to submit a payment and receive taxpayer's ID.</p> <p>User is expected to enter the following information:</p> <ul style="list-style-type: none"> • Taxpayer name (Alphanumeric) • Identifier Type (Lookup) • Identifier Number (Alphanumeric) • Email Address • Address Line 1 (Alphanumeric) • Address Line 1 (Alphanumeric) • City (Alphanumeric) • County (Alphanumeric) • State (Lookup) • Zip Code (Alphanumeric) • Paying On Behalf Of Somebody Else? (Boolean) • Identifier Type (Lookup) • Identifier Number (Alphanumeric) • Email Address (Alphanumeric) <p>Identification Requirements: none. Process synchronously, web service.</p>
IDENTIFY_TAX_CLEARANCE - Taxpayer Identification for Tax Clearance	<p>Provides the ability to validate taxpayer POIs required to request tax clearance certificate and receive taxpayer's ID.</p> <p>User is expected to enter the following information:</p> <ul style="list-style-type: none"> • Taxpayer name (Alphanumeric) • Date of Birth (Date) • Taxpayer Type (Lookup) • Identifier Type (Lookup) • Identifier Number (Alphanumeric) <p>Identification Requirements: none. Process synchronously, web service.</p>
TECHN-ISSUE – Report a website-related problem	<p>Allows user to notify revenue authority about technical issues with the website.</p> <p>User is expected to enter the following information:</p> <ul style="list-style-type: none"> • Taxpayer name (Alphanumeric) • Email Address (Alphanumeric) • Technical Issue – Topic (Lookup) • Detailed description of the problem (Text) <p>Identification Requirements: none. E-mail.</p>

Messages

The following messages are defined to support service request functionality:

Service Request	Description
TAX_CLEARANCE_CERT	15010, 15011, 15012
REFUND_STATUS_UNR	13001,13002, 13003,13010,13011,13012,13014,13015,13016,13017
IDENTIFY_TAX_PAYMENT	11001, 11002, 11003
IDENTIFY_TAX_CLEARANCE	15001, 15002

Configuration

In order to enable the supported service requests you may add/modify the configurations described in the following topics.

Service Request Config

Processing

Specify a recipient for each service request delivered via email.

Appearance

The service request description appears on the initial selection dropdown. You can change the overall tone of your dialog with the taxpayer by re-phrasing the descriptions of the generic service request(s) and the descriptions of request categories (lookup **SRCAT**).

Example:

Provided description: *Tax Clearance Certificate Request*

Override description: *Apply for a tax clearance certificate*

The service request configurations are provided in the base product without **Header**, **Footer**, and **Help**. Consider adding these to the service requests. Header and footer are displayed on the main service request input screen. The help is displayed on the Service Request Selection screen when the taxpayer picks the topic from the dropdown. You can enrich user's experience and communicate additional information, such as notes, disclaimers, explanations, etc.

Example:

Load the service request **TECHN-ISSUE**.

Press the **Preview** button to view the request's screen.

Now switch to the **Appearance** tab and enter the following:

Header:

```
<span style="font-family: 'Arial'; font-weight: bold; font-style: normal; text-decoration: none; font-size: 12pt; color: #666699;">Report technical problems to the Webmaster</span> </br> </br>
<span style="font-family: 'Century Gothic'; font-weight: normal; font-style: normal; text-decoration: none; font-size: 9pt; color: gray);">
Use this email service to report anything on this website that isn't working correctly or to make comments or suggestions for improving our website.
</span>
```

Footer:

```
<span style="font-family: 'Century Gothic'; font-weight: normal; font-style: normal; text-decoration: none; font-size: 9pt; color: gray);">
<font color=RED> NOTE </font> : This email form is <font color=RED> NOT SECURE. </font>
</br>
```

For your security and privacy, please do not include your social security number or other personal, confidential information in your message.

Help:

Request assistance finding something on the site, understanding our different file formats,
 printing files you've downloaded, installing or using the tax products CD-ROM, or any similar technical problem.

Press the **Preview** button to see the result.

Field Level Help

You can add the help text for each input field on the dynamic screen. The help text appears as the callout when the focus is moved into this input field.

Field Validation

The Fields provided in the base product have no Validation Rules attached. Add validation rules to enforce format rules on service request's input.

Example:

How to validate a phone number using US standard phone format:

- Add new message with the text, 'Invalid Phone Number'.
- Add new Validation Rule PHONE_NBR_US.
- Specify the newly added message. Enter the following validation expression:

```
^\(\(?([0-9]{3})\)\)?[- ]?([0-9]{3})[- ]?([0-9]{4})$
```

This expression validates the US phone pattern.

Entry may start with parentheses (optional) followed by three numeric digits, followed by a dash or space, followed by three numeric digits, followed by an optional dash or space, followed by four numeric digits.

Load the Service Request **TAX_CLEARANCE_CERT**.

Go to the Fields tab and find the field **PHONE_NBR**. Attach the new Validation Rule to the field.

Now try to enter an invalid phone number when submitting the request for Tax Clearance Certificate.

Lookup

You can add values for the Lookup: **RECIPIENT**. The values of this lookup are used to specify a recipient of an email service request. Use the extended value to specify recipient's e-mail address.

You can also add values for Lookups referenced by service request fields such as:

- **TAXCLRRSN** – Tax Clearance Reason
- **TAXPAYERTYPE** – Taxpayer Type
- **IDTYPE** – Identifier Type
- **STATE** – State
- **TAXTYPE** – Tax Type

Note: You can deactivate any/all of the lookup values provided with the base product.

Example:

Refund status request **REFUND_STATUS_UNR** includes a field **TAX_TYPE**. This Field is referencing a Lookup TAXTYPE. No values are provided with the base product. Decide what tax types should be available for the taxpayer paying for the filing period. Add these tax types as new lookup values.

Messages

Several info/error messages are provided for each service request (see "Supported Service Requests" for more information). You can add the DVM mapping for the messages you anticipate receiving from the revenue management system. You can also define new messages and use them instead of those provided with the base product.

Example:

Message 15001 "Your request has been processed; a Tax Clearance certificate was issued on {1}." is provided with the base product.

Scenario 1: The revenue management system returns message category 1/message number 1111 with one parameter of type Date upon successful processing of tax clearance certificate.

Use DVM OTSS_MessageNumber to map 15001 to 1:1111.

Scenario 2: The revenue management system returns message category 1/message number 1111 with no parameters upon successful processing of tax clearance certificate.

Override the message 15001 text, remove the substitution parameter and use DVM OTSS_MessageNumber to map 15001 to 1:1111.

Scenario 3: The revenue management system may return multiple messages: message category 1/message numbers 1111, 1112 and 1113 with parameters upon successful processing of tax clearance certificate.

Create brand new messages NNNNN and XXXX and use DVM OTSS_MessageNumber to map as follows:

```
15001 to 1:1111, NNNNN to 1:1112, XXXX to 1:1113
```

Validation Rules

You can add client-side validation to the service request fields. Regular expressions provide a powerful search pattern language and can be used to implement complicated business rules.

Examples:

EIN – US Employer ID Number: `^[0-9]{2}-[0-9]{7}$`

EIN Pattern: Starts with 2 numeric digits, followed by a dash(-), then ends with 7 numeric digits.

US Phone Numbers:

`^\\(?([0-9]{3})\\)?[-]?([0-9]{3})[-]?([0-9]{4})$`

Phone Pattern: May start with parenthesis (optional) followed by 3 numeric digits, followed by dash or space, followed by 3 numeric digits, followed by optional dash or space, followed by 4 numeric digits.

Properly formatted email address:

`^[a-zA-Z0-9_-]+(\\. [a-zA-Z0-9_-]+)*@[a-zA-Z0-9]+(\\. [a-zA-Z0-9]+)*(\\. [a-zA-Z]{2,})$`

Email Pattern: Starts with Alphanumeric characters [a-zA-Z0-9] or dash(-) or underscore (_), then maybe optionally followed by dot (.) plus Alphanumeric characters [a-zA-Z0-9] or dash(-) or underscore (_) then followed by @, then followed by a group of alphanumeric characters then maybe optionally followed by dot (.) plus alphanumeric characters (in case email has two levels, e.g., .com .ph), then followed by dot (.) plus characters having length of 2 or more.

Advanced Navigation

You can trigger a specific service request invocation from an HTML content using the following syntax:

```
/faces/oracle/webcenter/portalapp/pages/TaxpayerSvcReq.jspx?serviceReq=<put your  
Service Request Type Code here>
```

BPEL DVM Mapping

OTSS_ServiceRequestType

An entry is provided for each payment destination included in the base product. The column **OTSS_SRType** contains service request codes.

For each entry specify the corresponding translation values from your revenue management system in the column **EXT_SRType**.

If your revenue management system does not use the service request code, leave the translation value blank. As a result, the **<serviceRequestType>** node on the request xml would be empty.

If your implementation in the revenue management system designed to use the same request type codes as in web self service portal application, delete the records from the DVM. As a result the value in the **<serviceRequestType>** node on the request xml would be delivered 'as is'.

Examples:

The content of the node:

```
<serviceRequestType>TAX_CLEARANCE_CERT</serviceRequestType>
```

- With translation value **XXX** for **TAX_CLEARANCE_CERT**

```
<serviceRequestType>XXX</serviceRequestType>
```

- With blank translation value for **TAX_CLEARANCE_CERT**

```
<serviceRequestType></serviceRequestType>
```

- Without an entry in the DVM for **TAX_CLEARANCE_CERT**

```
<serviceRequestType>TAX_CLEARANCE_CERT</serviceRequestType>
```

OTSS_FieldCodes

You can translate the values of the lookups referenced by service request fields. Enter the field name and value pairs into the **OTSS_FieldNameValue** column in a format "FIELD": "VALUE". Enter the corresponding value from your revenue management system in the **EXT_FieldValue** column. This translation is optional.

Examples:

The content of the entry on **<destinationDetails>** list for the request field **TAX_TYPE** with value **IND** selected from the lookup **TAXTYPE**:

```
<requestField>
  <sequence>
    <fieldName>TAX_TYPE</fieldName>
    <fieldValue>IND</fieldValue>
  </requestField>
```

- With translation value **XXX** for **TAX_TYPE:IND**

```
<requestField>
  <sequence>
    <fieldName>TAX_TYPE</fieldName>
    <fieldValue>XXX</fieldValue>
  </requestField>
```

- With blank translation value for **TAX_TYPE:IND**

```
<requestField>
  <sequence>
    <fieldName>TAX_TYPE</fieldName>
    <fieldValue></fieldValue>
  </requestField>
```

- Without an entry in the DVM for **TAX_TYPE:IND**

```
<requestField>
  <sequence>
    <fieldName>TAX_TYPE</fieldName>
    <fieldValue>IND</fieldValue>
  </requestField>
```

OTSS_MessageNumbers

You can map the expected return message numbers from the revenue management system to the messages defined in self service portal application.

How To Enable New Web Service-based Request

Design considerations

Determine what information your revenue management system requires in order to provide the new service. It is usually one or more data items, including dates, amounts. You can also offer a selection of pre-defined answers (lookups). Decide whether the free-form text (comments, feedback, etc.) should be a part of the request. Define what input data is required and what is optional. Design the additional content (help, header and footer) for the user interface.

Determine whether taxpayer's identity should be verified prior to providing this service. This will define what Identification Request you will be using. One identification request **IDENTIFY_TAX_CLEARANCE** is provided with the base product. You may use it or create your own.

You may decide that a standalone identification step is not required and incorporate taxpayer identification details into the service request fields' collection.

Determine how you want this request to be handled by the communication layer. Consider the anticipated volume of the requests and also the actual effort required to process the request in the revenue management system. Choose to deliver the request synchronously or asynchronously.

Decide under which category this request follows. If no appropriate service request category exists, define a new one (add a new value to the lookup **SRCAT**, specify extended value **GENERAL**).

Decide how the taxpayer would invoke this service request. Several methods are available:

- New active service request will automatically appear on the service request selection page, request types dropdown.
- You can incorporate the link to the specific request in the web site content, using the Advanced Navigation techniques.

Configuration

Configure the service request according to the design.

In the SOA/BPEL layer, add the mapping for a service request type code and specific fields (if applicable). Also add the mapping for the response message.

Implementation

The revenue management system has to be able to interpret this service request's set of input fields, provide the appropriate service and return a confirmation message(s) under **<confirmationData>** or an error message under **<errorMessage>**.

How to Enable a New Email Service Request

Design considerations

Design the input required for this particular service and the identification requirements, similar to the generic service request. Use header, footer, or help to provide various guidelines, explanations, disclaimers, etc. You can also advise a taxpayer that sensitive private data should not be entered in the text area.

Configuration

Configure the email recipient for this request. Configure the service request itself according to the design, specify processing method **Email**.

How to Enable a New Identification Request

Design considerations

Determine what information your revenue management system requires in order to identify a taxpayer. Decide whether the free-form text (comments feedback etc.) should be a part of the request. Define what input data is required and what is optional. Design the additional content (help, header and footer) for the user interface.

Determine what information should be masked during the input.

This request should be delivered synchronously.

Configuration

Configure the service request according to the design.

In the SOA/BPEL layer add the mapping for a service request type code and specific fields (if applicable).

Implementation

The revenue management system has to be able to interpret this service request's set of input fields, identify a taxpayer(s) and provide the collection of the taxpayer IDs in the response <responseDetails> or an error message under <errorMessage>.

How to Enable a New Refund Status Request

Design considerations

Determine what information your back-end system requires in order to determine the refund status, similar to the generic service request. Define what input data is required and what is optional. Design the additional content (help, header and footer) for the user interface.

Determine whether a taxpayer's identity should be verified prior to providing this service. This will define what Identification Request you will be using. One identification request **IDENTIFY_TAX_CLEARANCE** is provided with the base product. You may use it or create your own.

You may decide that a standalone identification step is not required and incorporate taxpayer identification details into a service request fields' collection. The base product provides the refund request following this pattern.

The request should be processed synchronously.

Decide how the taxpayer would invoke this service request. Several methods are available:

- New active service request will automatically appear as the hyperlink on the **Where is My Refund?** (Refund Request Selection) page.
- You can incorporate the link to the specific request in the website content, using the Advanced Navigation techniques.

Configuration

You can configure the service request according to the design.

In the SOA/BPEL layer, add the mapping for the service request type code and specific fields (if applicable).

Implementation

The revenue management system has to be able to interpret this service request's set of input fields, determine the status of the refund, and return a confirmation message(s) under <confirmationData> and, optionally, refund amounts under <responseDetails>, or an error message under <errorMessage>.

FAQ: Service Request

This section includes frequently asked questions about Service Requests.

- **How do I define field validation on my Service Request?**

The basic data type and format validation can be defined using attributes of the Field entity. In addition, the Service Request Field can be marked as required.

You can create a new Validation Rule to implement more complicated validation logic using regular expressions.

Attach a Validation Rule to the Field (if the validation is always applicable) or to the Service Request Field, if the validation is applicable in a context of the specific service request.

Examples:

Scenario 1: A single Field is defined to represent US driver's license. There is only one valid format for the driver's license and it does not change. It is safe to assume that the same validation rule would be always applicable; therefore you can attach a Validation Rule to the Field.

Scenario 2: A single Field is defined to represent an Identifier Number. Multiple types of identifiers exist: a taxpayer ID, a driver license number, a professional license number and others and there is no universal formatting rule applicable for all of them. When this Field is used on a service request and supposed to represent one of the identifiers, you can attach request-specific Validation Rule to the Service Request Field.

- **Can the Service Request have multiple pages?**

This feature is currently not supported. However the implementation can model a business process as a sequence of several service requests.

The service request can be triggered from the HTML content (refer to [Advanced Navigation](#) for details), therefore a hyperlink to a service request can be embedded in the confirmation message text. This way, after submitting a service request successfully, the taxpayer may be "guided" to the next step in the process.

- **How can I enable hyperlink to a Service Request from different places on the web site?**

Refer to [Advanced Navigation](#) for the instructions.

- **When should the requirement be implemented as a tax or business registration form and when it can be modeled as a service request?**

The service request allows to model idiosyncratic scenarios beyond the form-based processing, such as:

- The revenue authority receives taxpayer's feedback on various topics.
- Taxpayer reports an issue with the website.
- Taxpayer requesting an informational package or paper documents from the revenue authority.
- Taxpayer is signing up for a program.

The service request framework can be used to address new business requirements instantaneously, "here and now", as opposed to the complex process of implementing a new tax or business registration form.

The service request interface is limited to a single screen. Therefore it is not feasible to use service requests for the functionality that requires the taxpayer to provide a lot of data.

Another aspect to be considered is the processing of the request in the revenue management system.

Scenario 1: A certain service is already available in the revenue management system. For example, there is a service that updates the account when the taxpayer is leaving the country for a prolonged time period. The input for the existing service is a taxpayer ID and the date range (start and end date). The requirement is to expose this service on the web self service portal.

This requirement is a good candidate for the service request-based implementation. It can be implemented using a service request with preliminary taxpayer identification.

Scenario 2: The change of taxpayer's mailing address should be reported using a specific form. The new address is the only information that needs to be collected from the taxpayer; the rest: previous mailing address, taxpayer name and identifiers is already stored in the revenue management system and can be derived.

This requirement can be implemented using a service request with preliminary identification. The web service may trigger the taxpayer details retrieval followed by the form creation.

- **How are request fields sent to the revenue management system?**

The XML node that represents the value of the service request field contains string.

The string value is formatted according to the service request Field's data type:

- String and Numbers are delivered 'as is'.
- Date Field's values are formatted according to the format specified in the system configuration options.
- Lookup Field's values are delivered 'as is' or may be translated in the BPEL layer.
- Boolean Field's values: true or false.

```
<serviceRequestData>
  <requestField type="list">
    <sequence>           //request sequence - as defined on service request
    <fieldName>         //field name - as defined on service request
    <fieldValue>        //field value - alphanumeric
  </requestField>
</serviceRequestData>
```

- **Can I control appearance of the request fields?**

The field is rendered on the screen according to its data type. It may appear as a date, with adjacent calendar, as a checkbox, as a multi-line text area or as a simple single-line input.

- **Can I include the comments and instructions about request fields?**

You can define a hint string on the Field. It will be displayed next to the input widget. The hint may simply suggest the format of the input. Another option is to define an extended help text. It will be displayed as a callout when user hovers over the field or when the field is in focus. The help defined on the Field entity will be displayed everywhere this Field is used. You can override this help for the specific service request by defining help on Service Request Field.

Chapter 5

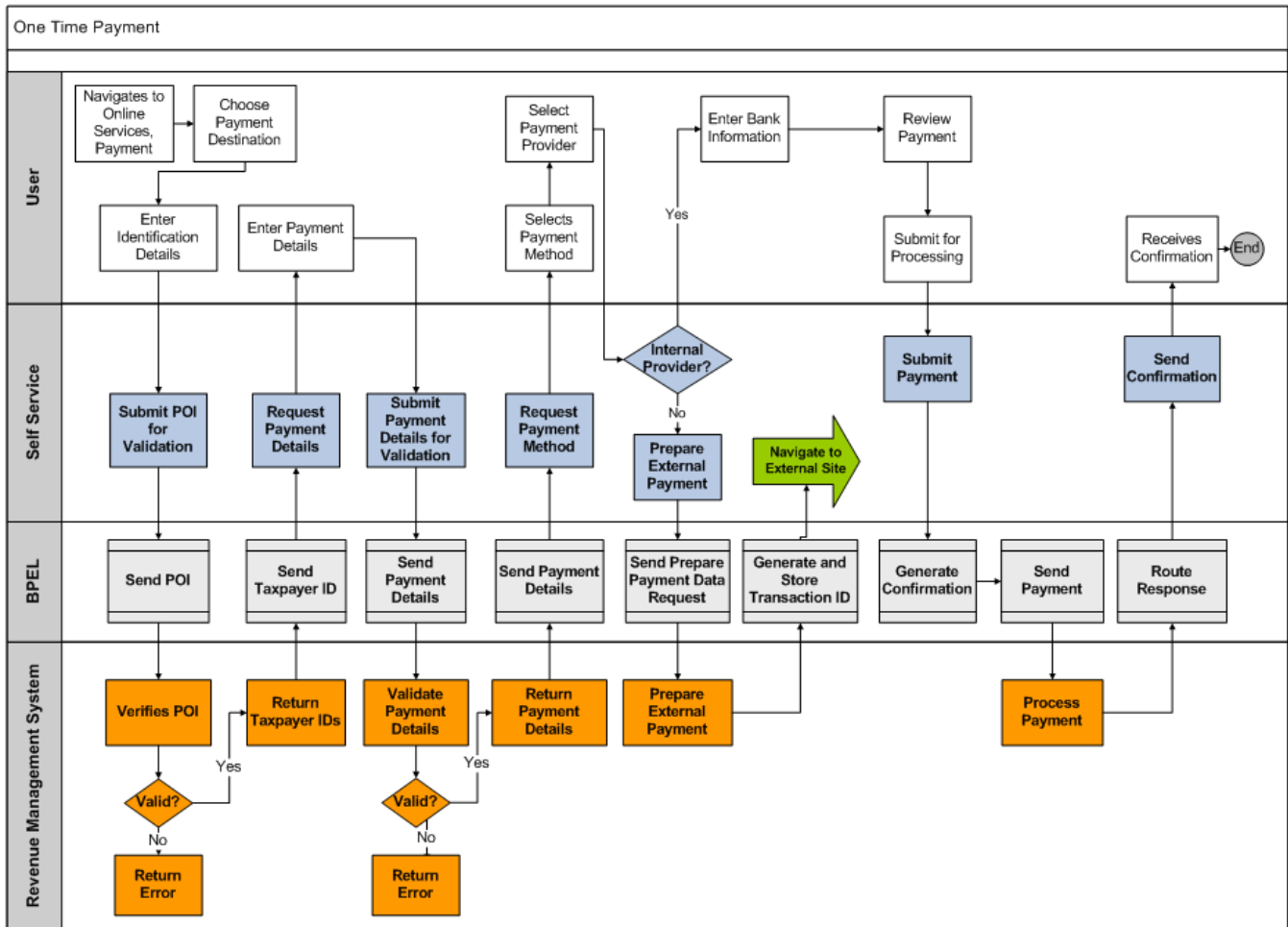
One-Time Payments

Overview

The Make a Payment option is accessible from the OnLine Services pull-down menu on the web self service portal navigation bar and from the left-side navigation panel.

The taxpayer does not have to register as a web self service user and/or login in order to make a payment. The taxpayer is prompted to provide identification details prior to making the payment.

Payment



Payment Option Selection

This screen displays the available payment options. The selection represents the list of active Payment Destinations configured in the system. The taxpayer selects one by clicking on the hyperlink.

Taxpayer Identification

The user is asked to provide Proof of Identity (aka POI) details before being allowed to make a payment. This step applies to casual users only.

POI requirements are defined on Payment Destination.

Note: Taxpayer identification is implemented using Taxpayer Identification service request. Service request code is derived from Payment Definitions, then the service request configuration is retrieved and the Taxpayer Identification screen is rendered dynamically.

The taxpayer enters the identification information.

Note: At this time, a web service **TSTaxpayerIdentification** is triggered and the collected input is sent to the revenue management system for verification. The successful response contains a collection of taxpayer Ids.

Payment Details

The user is prompted to enter payment amount and specific details describing this payment's purpose. For example, when paying a collection notice, user is asked to enter collection notice number. The required input for each specific payment option is defined on the corresponding Payment Destination.

Note:

At this point, a web service **TSPrepareExtPaymentData** is triggered with action **VALIDATEONLY**, and payment details are sent to the revenue management system for verification.

The input is verified in two layers:

- The front-end validates whether the required input values are provided and checks the data type.
- The revenue management validates the details according to the business rules.

Payment Method Selection

After payment details are validated, taxpayer selects a payment method: credit card, checking account etc. The list is derived from the lookup **TENDERTYPE**.

- **Scenario 1:** Selected Payment Method is supported by more than one Payment Provider or Payment Method is supported by a single external payment services provider. In this case the user is presented with selection of payment providers
- **Scenario 2:** Selected Payment Method is supported by a single internal payment provider. In this case the user is prompted to enter payment method details

Payment Method Details

If the payment method is supported by a single internal payment provider, **Bank Account Information** must be provided. The user is prompted to enter bank routing number and account number.

Selection of Payment Providers

If the payment method is supported by more than one payment provider, the list of payment providers is displayed. Each entry on the list contains the provider's description (hyperlink) and an explanation (help text derived from the payment provider's configuration).

Payment Review

All previously collected information is presented to the user for review. User may either submit the payment or abort the transaction.

Note: At this point, a web service **TSTimePayment** is triggered and the collected input is sent to the revenue management system for payment creation.

Confirmation

The expected web service response contains the Confirmation number and message(s). This information is displayed to the user. The confirmation may be printed for later reference.

Payment with External Provider

The flow follows this branch when user selects to pay with an external provider. It starts from **Payment Method Selection** page.

Selection of Payment Providers

The taxpayer selects a provider from the list by clicking on the hyperlink.

Note:

At this point, a web service **TSPrepareExtPaymentData** call is triggered to retrieve additional payment details. The response expected contains the information in a form of name/value pairs. This information may include taxpayer's mailing address, e-mail etc. - according to Payment Provider's specifications. The response may also include the indication whether taxpayer is expected to pay a convenience fee.

If the web service call was successful, BPEL process assigns a unique ID to this transaction and registers it in the internal table. This ID is included in web service response message.

Redirection To External Website

After retrieving external payment details the system invokes a plug-in defined on the provider record. Payment details are formatted and user is redirected to the provider's website with an HTTPS post. Refer to [Redirect Payment Process to Official Payments](#) for more details.

Paying on External Web Site

Taxpayer provides the required information and completes the payment.

Receiving External Payment Data

External provider transmits the information about completed payment. The transportation methods and means may vary from one provider to another. It could be an HTTP/HTTPS post or a web service. The post back message typically contains a transaction reference ID (i.e. credit card authorization code) that may be used to trace this payment in all systems involved in payment process: web self service/revenue management application, payment provider and the credit card company or bank.

Payment Report and Reconciliation

The typical integration with an external provider includes receiving and processing of a payment report. It is sent after this provider finalizes the payments, i.e. submits them to the credit card company. The web service **TSPProcessExtPayReportRecord** is designed to accommodate the processing of a single payment report record.

Note: The model integration with Official Payments Corporation (USA) is described below. In this integration, the XML message is sent as post back HTTP. It is received in the SOA layer and transformed by a BPEL process into a One-Time Payment web service.

Defining Payment Destination

Payment Destination defines what input the taxpayer is asked to provide in order to make a certain payment. In other words, Payment Destination describes "what I'm paying for".

Payment Destination Search

The user can search for an existing Payment Destination.

The standard **Edit** and **Delete** functions are available for each Payment Destination.

You can click the **Add** button to define a new Payment Destination.

Payment Destination Maintenance

Payment Destination - Main

Payment Destination is a user-defined code, an identifier of the entity.

Active indicates whether this Payment Destination is ready for use. The list of active payment destination is displayed on Payment Option Selection page.

Description is the hyperlink text displayed for this entry on the Payment Options selection page.

Identification Request dropdown presents a selection of taxpayer identification service requests. At run-time this identification request is invoked and a taxpayer is prompted to enter the Proof Of Identity (POI) information. Identification is required only for casual (not signed in) web site users.

Appearance

Help appears on the Payment Details page beneath the payment destination fields. It can be entered as HTML or as plain text.

Payment Destination - Fields

This tab displays the Payment Destination's **Fields** list. Fields represent the information that should be collected from the taxpayer in order to process this specific type of payment.

The standard **Edit** and **Delete** functions are available for each field on the list.

You can click the **Add** button to define a new payment destination field.

Payment Destination Field

Payment Sequence uniquely identifies the field in the list. It is copied to the web service request paymentDestinationDetails collection and sent to the revenue management system.

Field is referencing a Field entity. By default, the properties of this Field are used to render the input UI.

Display Sequence controls the order of the input fields on the screen.

Required indicates whether the input value is required or optional.

Display Description provides an alternative label for the field on the screen.

Display Help provides an alternative popup help text for the field on the screen.

Example:

Consider the following scenario: One of the required inputs for your Payment Destination is a taxpayer's legal name and you also have to explain the term 'legal name'. An existing Field, **ENTITY_NAME** has all the attributes you need, except it has a description 'Name', but no help. You can specify Display Description '**Legal Name**' and add the necessary explanation using the **Display Help**.

Defining Payment Provider

The Payment Provider definition is used to orchestrate the final steps in online payment submission. It represents the application (the revenue management system or an external payment service) where the payment is sent for processing.

Payment Provider Search

the user can search for an existing Payment Destination.

The standard **Edit** and **Delete** functions are available for each Payment Provider.

You can click the **Add** button to define a new Payment Provider.

Payment Provider Maintenance

Provider is a user-defined code identifying the entity in the system.

Active indicates whether this Payment Provider is in use. The list of active payment destination is displayed on Payment Option Selection page.

Description appears as a hyperlink text when the user is presented with the list of available providers.

Navigation Style defines whether user is redirected to provider's web site or remains on the portal page. Valid values are: Internal and External.

Prepare Data Plug-in is a module (Java) used to prepare information to be sent to the external web site. Refer to the [Preparing the Data Plug-in](#) section for detailed technical information.

This field should contain full qualified name (e.g., the package name) for the Java class that would be used to process the payment details for an external payment provider. This field is required when Navigation Style for the payment provider is External. If the taxpayer chooses to make a payment via external provider, the web self service application instantiates the class referenced in this field.

Note:

Integration with Official Payments Corporation is provided with the base product. When creating payment provider metadata for Official Payment Corporation, choose the plug-in `oracle.apps.otss.payment.txn.ui.bean.opc.OfficialPaymentBean` (or implement your own class).

The base product also provides a sample Prepare Data plug-in extension class (`'oracle.apps.otss.extension.payment.SamplePaymentExtension'`).

Redirect URL points to the external provider's web site location.

Help can be used to show the information about this provider when the user is presented with the list of available providers. May be entered as HTML or as plain text.

The **Supported Payment Methods** section shows all methods available in the system. The list is derived from the active values of the lookup **TENDERTYPE**.

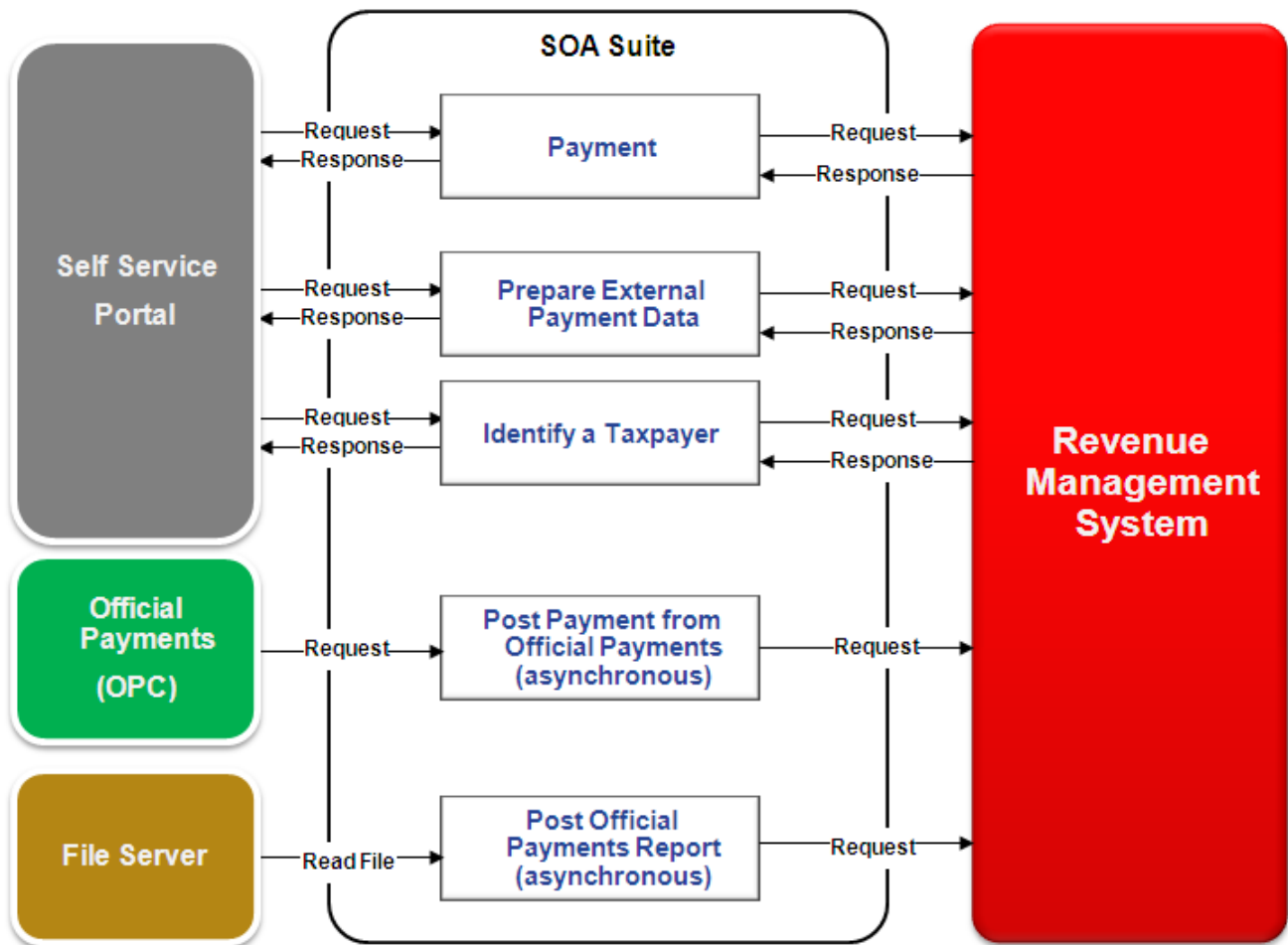
On the Payment Selection Page the taxpayer selects a payment method from the dropdown, and all active providers that support this method are displayed.

Supported Payment Providers

The base product provides definitions for a single Payment Provider. This entry represents the revenue management system itself. Its definition cannot be modified by the implementation, although this provider can be deactivated.

You may need to configure an additional (external) Payment Provider as part of the implementation of the model integration with Official Payments Corp.

BPEL Processes



The integration SOA composites invoke the revenue management system web services and send the response back to self service portal application.

A confirmation number is generated when payment service is invoked. This confirmation number is returned as part of the response.

For payments processed by Official Payments Corp, the integration patterns are:

- For payment posting, Official Payments Corporation sends an XML post back message to the SOA composite, which after processing, sends the payment web service request to the Revenue Management System.

- For payment report posting, Official Payments Corporation sends the report file to a file server and the SOA composite reads the file and sends individual records in the file to the Revenue Management System as web service requests.

The common features of all provided processes are as follows:

- SOA Composite processes have pre-transformation extension point and post-transformation extension points as explained in section Integration Extensibility.
- SOA Composite processes have extension templates for each level of the outgoing message XML schema as explained in section Integration Extensibility.
- All endpoint invocation are using dynamic partnerlink URL and are configurable.
- For confirmation numbers, the Confirmation Number Utility service is called using dynamic partnerlink.

Refer to the [BPEL/SOA Integration](#) chapter for detailed information about integration layer implementation

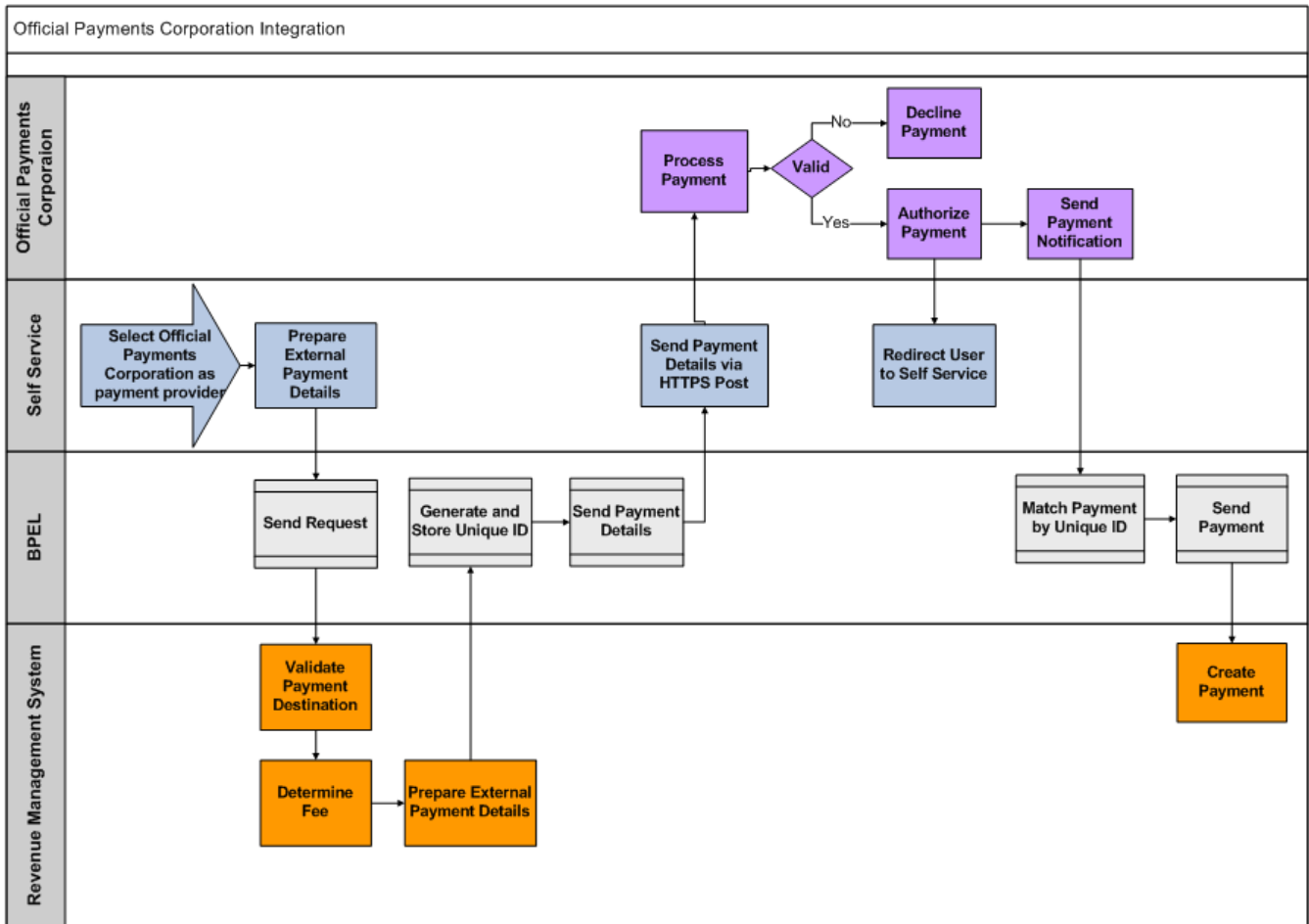
Integration with Official Payments Corp.

This section provides information related to the product's model integration with Official Payments Corporation as an external payment provider.

OPC Integration Overview

Official Payments Corp. provides support for multiple payment vehicles: credit and debit cards, electronic checks, Bill Me Later[®] and Green Dot Money Pack[®].

OPC Integration Process Flow



The model integration provided with the base product allows the taxpayer to select a payment option ("what am I paying for?"), enter appropriate details and amounts on the web self service portal, and have this information validated. The taxpayer is then redirected to the Official Payments web site to complete the payment using a credit card.

Credit card payments are verified/authorized immediately and the taxpayer receives the confirmation number and e-mail confirmation from Official Payments. This same confirmation number is transmitted to the revenue management system with an XML post-back.

Official Payments submits the verified payments to the final target (credit card company) and provides daily reports for further reconciliation.

This integration features a SOA composite process that reads Official Payments Corp. report (flat file), performs the transformation and issues web service calls to the revenue management system.

Redirect Payment Process to Official Payments

The last step before the redirection is a web service is to prepare the information to be sent for the HTTP Post.

Post-in Parameters

Official Payments expects to receive the following data with the HTTP post: Required:

- Product ID (**productId**) – the unique ID generated by Official Payments for each payment scenario/ configuration. For this integration Official Payments created two Products: one for the case where the taxpayer is paying a payment handling convenience fee and another for the case where this convenient fee is absorbed by the tax authority.
- Unique ID (**cde-UniqID-0**) - the unique identifier for the transaction.

Optional – predefined names:

- paymentAmount
- firstName
- middleName
- lastName
- suffix
- address1
- address2
- cityName
- provinceCd
- postalCd
- phoneNum
- email
- postbackUrl
- returnUrl
- errorUrl
- cancelUrl

In addition, the following custom data fields are configured for this model integration:

- cde-Field-1 - Payment Destination Details, first field
- cde-Field-2 - Payment Destination Details, second field
- cde-Field-3 - Payment Destination Details, third field
- cde-Field-4 - Payment Destination Details, forth field
- cde-PaymOpti-5 - Payment Destination
- cde-RefeLine-6 - Payment Destination Description (displayed on Official Payments screen)
- cde-RefeLine-7 - Concatenated, comma-delimited values of Payment Destination fields (displayed on Official Payments screen)
- cde-BusiName-8 - Leave blank, at the moment this information is not available (displayed on Official Payments screen)

Receiving Payment Details

The values for the post-in parameters are retrieved from the revenue management system using web service **TSPrepareExtPaymentData**. The request contains all the information collected from the taxpayer up to this point: payment destination and destination fields, payment amount and also the taxpayer ID(s) and the Payment Provider code. The response from the revenue management system contains the payment details list in the form of name/value pairs:

```
<paymentDetails>
  <sequence>
    <fieldName>
      <fieldValue>
```

</paymentDetails>

If none are retrieved, the Prepare Payment Data for Official Payments plug-in (described below) populates the required parameters and Custom Data Elements on the HTTP Post.

Prepare Payment Data for Official Payments

The plug-in provided with this integration, `oracle.apps.otss.payment.txn.ui.bean.opc.OfficialPaymentBean`, performs the following:

1. It implements **prepareData()** method of the interface and builds the list of name-value pairs that needs to be send to Official Payments website with the Form Post.
2. Adding the following elements to the list:
 - Name: `cde-UniqID-0`
 - Value: the `externalId` value. Is there is no external id provided, the value added as "
 - Name: `cde-PaymOpti-5`
 - Value; payment destination code. If no value available, " is added.
 - Name: `cde-Refeline-6`
 - Value: payment destination's description
 - Name: `productId`
 - Value: derive the extended value corresponding to the **fee requirement** value from the lookup **OPCPRODUCT**. System throws an error is product id is not found.
 - Name: **editAmount**
 - Value: derived from the System Configuration Option, *true* or *false*
 - Names: **returnUrl**, **cancelUrl**, and **errorUrl**
 - Values: derived from the corresponding System Configuration Options
 - Name: **redirectURL**
 - Value: derived from the Payment Provider definition. If not found, the system error is thrown.
3. Add payment details as below:
 - The contents of the **paymentDetails** collection are added to the list of elements.
 - If no payment details were received from the revenue management system, the following entries are added:
 - Name **paymentAmount** Value: payment amount
 - Name **cde-BusiName-8** Value: "
 - Names **cde-Field-1**, **cde-Field-2**, **cde-Field-3**, and **cde-Field-4**
 - Values: payment destination fields with sequence 1,2,3 and 4 respectively
 - Name: **cde-Refeline-7** Value: concatenated value of payment destination fields, comma-delimited

The **prepareData()** method returns the list of this elements to the handler that does further processing.

Limitations

The provided configuration of Custom Data Elements supports up to **four** payment destination fields.

Post-back XML Processing

The SOA composite process **OTSSPaymentPostingRequestEBF** receives the post-in XML. It is filtering incoming post-in XMLs and further processes only successful payment submissions, with `<resultCode>A</resultCode>`, meaning "Approved".

Transaction verification

The post-back XML contains the unique transaction/session identifier generated in web self service and transmitted to the Official Payments on the post-In XML.

The logic in SOA composite process **OTSSPaymentPostingRequestEBF** verifies whether the record with this identifier exists in the internal table. This is an additional security measure to prevent fraudulent payment submissions.

Payment Creation

The post-in XML is transformed and submitted to the revenue management system using **TOneTimePayment** web service.

The **post-back XML** contains all the data transmitted with the **post-in XML**, including all custom data elements.

It provides the payment destination code and destination details (values only, no field names), payment amount, date, and payment type (payment method) and more. The extra information, specific for Official Payments is placed under the `<externalPaymentData>` node.

Note: Official Payments have their own codes to represent various payment methods (referenced as **account types** in Official Payments documentation) : credit cards, e-checks etc. The dedicated DVM **OPC_PaymentType** is provided in order to translate these codes into revenue management system's values and populate `<paymentType>` element.

It also contains two unique IDs:

- The **transaction/session identifier** is transmitted to the revenue management system web service as `<externalId>`.
- The **transaction authorization code** displayed to the taxpayer as a confirmation number on the Official Payments site is transmitted to the revenue management system as `<extTransactionReferenceId>`.

Confirmation Number for Official Payments

When taxpayer completes his credit card payment with Official Payment Credit Card processor he receives a confirmation message. This message is produced by Official Payment and contains confirmation ID generated by it.

Self service rather than generating the confirmation number uses the number provided by Official Payment. To make this number unique in revenue management system, BPEL process adds a prefix to this number. (See the "SOA Composite OTSSPaymentPostingRequestEBF" description for more details.)

If taxpayer wants to track the payment using the confirmation number provided by Official Payment he would also need to add the same prefix.

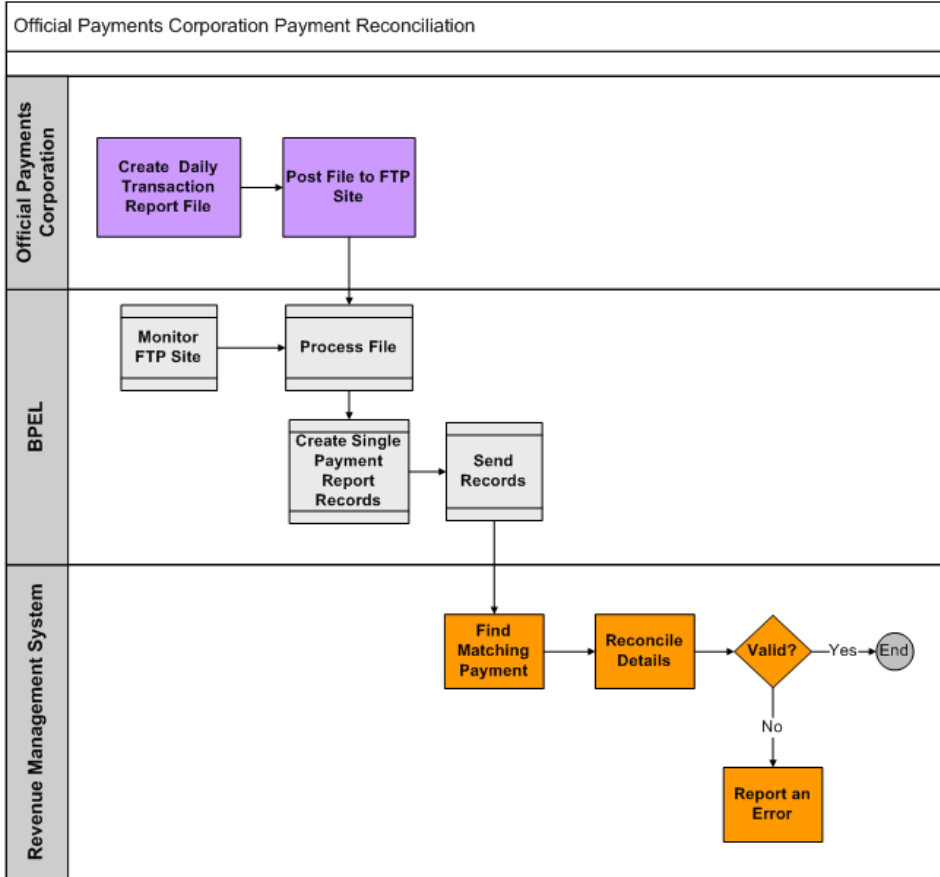
Implementation should provide the instructions on Track Your Transaction UI that would explain the user how to make this concatenation.

Implementation may use the following methods to advise the taxpayer about this option:

- Include the explanation about these special confirmation numbers in the in-line help on the **Online Services - Track Your Transaction** page.
- Include the explanation in the help when you configure a Payment Provider entry for Official Payments.
- Create a new portal page with HTML (or other) content using Web Center Composer and enter the explanation there. Specify this page as the **OPC_RETURN_URL** system configuration option.

Payment Report Processing

Payment Report Process Flow



Official Payments provides the payment report file for daily credit card and electronic check transactions, and returned electronic check transactions.

This comma-delimited file will be transmitted from Official Payments via e-mail or made available via FTP, on a business day basis Monday through Friday including bank holidays.

SOA composite process **OTSSReportReconciliationRequestEBF** reads the file and submits payment report records to the revenue management system for reconciliation using web service **TTSPProcessExtPayReportRecord**.

Each report record contains two unique identifiers:

- The **transaction/session identifier** generated in web self service and included in both post-in and post-back XML-s and transmitted to the revenue management system with **TSOneTimePayment** web service as **<externalId>**.
- The **transaction authorization code** displayed to the taxpayer as a confirmation number on Official Payments site. It is included in the post-back XML and transmitted to the revenue management system with **TSOneTimePayment** web service as **<extTransactionRefID>**.

Note:

The assumption is that the payment created in the revenue management system can be traced by either one (or both) identifiers.

The reconciliation logic should be triggered by the **TTSPProcessExtPayReportRecord** web service request.

Official Payments Integration Configuration

Payment Provider

Add a Payment Provider that represents Official Payments Corporation.

Specify oracle.apps.otss.payment.txn.ui.bean.opc.OfficialPaymentBean as the Prepare Data Plug-in.

Specify Official Payments testing site https://staging.officialpayments.com/pc_entry_cobrand.jsp as Re-direct URL. Note that this is a temporary definition and should be changed later to point to Official Payments production site.

Define the Help contents (as an HTML or a plain text).

Indicate Credit Card as a supported Payment Method.

Convenience Fee Indicator

For this model integration Official Payments has defined two "products", tentatively assigned descriptions **Taxes1** and **Taxes2**:

- Product ID 27362262501965139940530820135778488 for payments whose convenience fee is absorbed by the tax authority.
- Product ID 481936748383120208732647342052473017 for payments whose convenience fee is paid by the taxpayer.

The Product IDs are stored in OPCPRODUCT lookup as extended values:

- NFEE - 27362262501965139940530820135778488
- CNVF - 481936748383120208732647342052473017

BPEL DVM Mapping

OTSS_FeeRequirement

Add an entry in the column **EXT_FeeReq_PayVendor_PayDestination** for every combination of:

- Convenience fee indicator from your revenue management system:
 - Payment Provider (code) for Official Payments
 - Payment Destination (code) from your revenue management system, delimited by ":"

For each entry, specify the corresponding Official Payment Fee Requirement value defined in the web self service application (active **OPCPRODUCT** lookup values) in the column **OTSS_FeeRequirement**.

Example:

For provider OPC, fee requirement **false** and payment destination **XXX**, the entry would be:

```
OTSS_FeeRequirement      EXT_FeeReq_PayVendor_PayDestination
NFEE                     false:OPC:XXX
```

OPC_PaymentType

Map account types provided by Official Payments Corp. The base product includes the entry for each Official Payments Corp. account type in the column **OPC_PaymentType**.

For each entry specify the corresponding translation values from your revenue management system in the column **EXT_PaymentType**.

Customizing the Integration With Official Payments Corporation

Official Payments Products

A customized setup is made by Official Payments Corp. for every implementation. Official Payments creates one or multiple products that represent different payment processing scenarios on their web site.

The provided model integration uses two products: one for convenience fee paid by the taxpayer and another for convenience fee absorbed by the tax authority. You may request different product setup from Official Payments. For example your business requirements may include discounted convenience fee.

The Product ID is included in the HTTP Post. The logic to determine the appropriate Product ID is performed by Prepare Data java plug-in.

Extend Convenience Fee Determination Logic

You can configure various levels of the convenience fee based on the payment destination without changing the implementation of **TSPPrepareExtPaymentData**.

Example:

The requirement is to charge the discounted convenience fee for the timely filing period payments.

Assuming that the **<convenienceFee>** node on **TSPPrepareExtPaymentData** response contains *true* or *false*.

Request to setup a separate Official Payments product 999999999 for the discounted fee. Add a new value **DSCF/99999999** to the **OPCPRODUCT** Lookup. Using the mapping in **OTSS_FeeRequirement**, assign this value associated with the discounted fee to this payment destination:

- Enter **DSCF** in the **OTSS_FeeRequirement** column.
- Enter **true:OPC:FILE_PERIOD** in the **EXT_FeeReq_PayVendor_PayDestination** column.

Payment Methods

In addition to the credit card payments you can enable e-check, debit card and other payment process methods via Official Payments.

Add an entry to System Configuration Options as follows:

- Configuration Option **OPC_EDIT_AMOUNT**
- Option Value Type: **Lookup**
- Lookup **Yes/No Indication:**
 - Set the value to **No** if only credit card payments should be made via Official Payments Corp.
 - Set the value to **Yes** to allow the taxpayer to choose from all the payment methods available via Official Payments Corp.

Update the Supported Payment Methods on the Payment Provider for Official Payments.

Redirect URLs

After completing the payment on the Official Payments Corporation website the taxpayer is redirected back to the web self service portal.

An implementation may customize taxpayer experience by either defining different redirection URLs or using the same URL so the taxpayer always returns to the same page on the web self service portal.

Using Web Center Composer you can create a new portal pages with HTML (or other) content and then add the following System Configuration Options:

- The page where the taxpayer is taken successful payment completion is defined in **OPC_RETURN_URL**.
- The page where the taxpayer is taken upon canceling the payment is defined in option **OPC_CANCEL_URL**.
- The page where the taxpayer is taken upon getting payment error is defined using option **OPC_ERROR_URL**.

Confirmation Number for Official Payments

The prefix for the confirmation numbers for payments made on Official Payments web site is configured in BPEL properties file. You can override the prefix.

Custom Data Elements

This integration is based on a pre-negotiated configuration of **Custom Data Elements** (see above).

- This configuration dictates the transformation and translation rules for SOA Composites that handle XML Post-back and the payment report file processing.
- The logic in provided Prepare Data Plug-in is built to support this configuration.

An implementation may negotiate different Custom Data Elements with Official Payments, re-implement **TSPrepareExtPaymentData** so that it retrieves different external payment details and create an alternative Prepare Data plug-in.

Implementing One Time Payment

The following sections provide the information needed to fully implement the one-time payment functionality provided with the product.

Web Services

In order to support the web self service payments the revenue management system is expected to implement the following web services:

Web Service	Description
TSSOneTimePayment	Based on the information collected from the taxpayer on the web portal, this should trigger the payment creation process and return a confirmation or error with the response.
TSPrepareExtPaymentData	This service is called twice with different value in the <action> node: VALIDATEONLY action. Called after the taxpayer entered the payment destination details and payment amount. It expected to validate this input and return an error, if invalid. PREPARE action. Called immediately before the redirection of the payment process to the external payment provider. The request contains payment provider code, payment destination code and details and payment amount. The response expected to include the convenience fee indicator and the collection of payment details.
TSPProcessExtPayReportRecord	The request includes the single payment report information. It expected to trigger payment reconciliation logic in the revenue management system. No response expected.

Prepare External Data Plug-in

This plug-in is used to finalize the list of payment details that will be sent to the payment services provider with HTTP form Post. It composes the collection of payment details (name/value pairs) according to provider's specifications.

The base product provides a default payment extension class (**oracle.apps.otss.extension.payment.SamplePaymentExtension**). This class can be used if the payment destination

details and external payment details retrieved by the Prepare External Data web service can be sent to the external provider's website 'as is'. For all other cases, the client must implement their own extension class.

The base product supports out-of-box integration with Official Payments Corporation. When creating the payment provider metadata for Official Payment Corporation, the client could choose to use the official class provided by Oracle (`oracle.apps.otss.payment.txn.ui.bean.opc.OfficialPaymentBean`), or could implement their own class.

Implementing the Payment Class

The payment class provided with the self-service product can be customized and/or extended through the `oracle.otss.extension` shared library.

To extend the sample payment class:

1. Using the `AdflibPaymentExtension.jar` library, create a new Java class by implementing **`oracle.apps.otss.extension.payment.PaymentExtension`**.
2. Implement the **`public List<Element> prepareData(PaymentDetail paymentDetail)`** method. The `paymentDetail` object provides all collected details, so this method should return a list of all the elements that must be passed, including the redirect URL.
3. Add the new class files to the `WSSExtension.war`.
4. Deploy the war as shared library:
 - a) Shut down the WSSPortal application.
 - b) Deploy `WSSExtension.war` as a shared library.
 - c) Restart or redeploy the WSSPortal application.

Payment Class Extension Sample Code

```
package oracle.apps.otss.extension.payment;

import java.util.ArrayList;
import java.util.List;
import oracle.apps.otss.payment.txn.extn.Element;
import oracle.apps.otss.payment.txn.extn.PaymentDetail;
import oracle.apps.otss.payment.txn.extn.PaymentExtension;

/**
 * This is the demo class that implements Payment Extension Interface. It will
 * implement method prepareData that would be called from the WSS payments
 * module for the payment provider who uses this class in 'Prepare Data Plug-
in'
 * attribute.
 */
public class SamplePaymentExtension implements PaymentExtension {
    public SamplePaymentExtension() {
        super();
    }

    /**
     * Implement this method as this will be called from the WSS application
     * when making payment.
     * @param paymentDetail : This method takes in object of PaymentDetail which
     * contains all the details for the payment which is either provided by the
     * user or collected from the backend application as response to collect
     * details call.
     * @return : return List of Element which needs to be send to
     * external URL.
     */
}
```

```

public List<Element> prepareData(PaymentDetail paymentDetail) {
    List<Element> dataList = new ArrayList<Element>();
    // Check for empty paymentDetail
    if(paymentDetail != null){
        // Add all the elements/data from the payment destination details.
        // Add all Payment Destination Fields.
        List<Element> destDetailList = paymentDetail.getDestDetailList();
        if (destDetailList != null && destDetailList.size() > 0) {
            for (Element e : destDetailList) {
                // Add Field Name and Field Value to the list.
                dataList.add(new Element(e.getName(), e.getValue()));
            }
        }
    }
}

```

Supported Payment Destinations

The following table lists the payment destinations supported by the product out of the box.

Payment Destination	Description
COLL_NOTICE – Pay a Collection Notice	User is expected to enter a Collection Notice ID, alphanumeric, up to 15 characters. Identification Requirements: IDENTIFY_TAX_PAYMENT (Identification For Payments).
PAY_PLAN – Pay a Payment Plan	User is expected to enter a Payment Plan ID, alphanumeric, up to 15 characters. Identification Requirements: IDENTIFY_TAX_PAYMENT (Identification For Payments).
FILE_PERIOD – Tax Type/Filing Period	User is expected to select a Tax Type from the dropdown (lookup TAX_TYPE) and also enter Filing Period Start and End dates. Identification Requirements: IDENTIFY_TAX_PAYMENT (Identification For Payments).

Messages

Messages defined for the One Time Payment functionality are:

11004,11005,11006,11007,11008,11009

Configuration

To enable the supported payment destination you may add/modify the Lookup, Payment Destination, and Messages configurations as described in the following topics.

Lookup

Add values for Lookups referenced by payment destination fields.

Example:

Payment destination **FILE_PERIOD** includes a field **TAX_TYPE**. This Field is referencing a Lookup **TAXTYPE**. No values are provided with the base product. Decide what tax types should be available for the taxpayer paying for the filing period. Add these tax types as new lookup values.

Payment Destination

Activate Applicable Entries

Decide whether you want to use any/all of the payment destinations provided in the base product. Inactive payment destinations do not appear on the Payment Options selection page.

Identification Requirements

Evaluate the Identification Request **IDENTIFY_TAX_PAYMENT** provided with the base product. Create another Identification Request if this one does not satisfy your business requirements.

Appearance

- You may override Payment Destination description, if needed.
- Add payment destination help (HTML or plain text). It will appear beneath dynamically rendered destination fields on the Payment Details page.

Example:

Load the payment Destination **COLL_NOTICE**. Enter the following Help:

```
<br />
<span style="font-family: 'Verdana'; font-weight: normal; font-style: normal; text-
decoration: none; font-size: 9pt; color: gray;">
Read your notice carefully – it'll explain how much you owe and how to pay it.
<br />
Pay the amount you owe by the date the notice asks. <br />
Make a payment plan if you can't pay the full amount you owe.<br />
Contact us if you disagree.<br />
</span>
```

Make an attempt to pay a collection notice and observe the help's appearance.

- Add field-level help for all or some of the payment destination Fields.

Messages

Several info/error messages are provided for each service request. Add the DVM mapping for the messages you anticipate to receive from the revenue management system. You can also define new messages and use them instead of those provided with the base product.

Example:

Message 11005 "Payment of {1} received on {2} will be posted on your account." is provided with the base product.

Scenario 1. The revenue management system returns **message category 1/message number 1111** with two parameters for successful processing of the payment.

Use DVM **OTSS_MessageNumber** to map **11005** to **1:1111**.

Scenario 2. The revenue management system returns **message category 1/message number 1111** with no parameters for successful processing of the payment.

Override message **11005** text, remove the substitution parameter, and use DVM **OTSS_MessageNumber** to map **11005** to **1:1111**.

Scenario 3. The revenue management system may return multiple messages: **message category 1/message numbers 1111, 1112 and 1113**, with parameters for successful processing of the payment.

Create new messages *NNNNN* and *XXXX* and use DVM **OTSS_MessageNumber** to map them:

```
11005 to 1:1111, NNNNN to 1:1112, XXXX to 1:1113
```

BPEL DVM Mapping

OTSS_PaymentDestination

An entry is provided for each payment destination included in the base product. The column **OTSS_PaymentDestValue** contains payment destination codes.

For each entry specify the corresponding translation values from your revenue management system in the column **EXT_PaymentDestValue**.

If your revenue management system does not use the payment destination code, leave the translation value blank. As a result, the **<payDestinationType>** node on the request xml would be empty.

If your implementation in the revenue management system designed to use the same payment destination codes as in web self service portal application, remove the records from the DVM. As a result the value in the **<payDestinationType>** node on the request xml would be delivered 'as is'.

Examples:

The initial content of the node is:

```
<payDestinationType>COLL_NOTICE</payDestinationType>
```

With translation value **XXX** for **COLL_NOTICE**

```
<payDestinationType>XXX</payDestinationType>
```

With blank translation value for **COLL_NOTICE**

```
<payDestinationType></payDestinationType>
```

Without an entry in the DVM for **COLL_NOTICE**

```
<payDestinationType>COLL_NOTICE</payDestinationType>
```

OTSS_FieldCodes

Translate the values of the lookups referenced by payment destination fields. Enter the payment destination field name and value pairs into the **OTSS_FieldNameValue** column in a format "**FIELD**":"**VALUE**". Enter the corresponding value from your revenue management system in the **EXT_FieldValue** column. This translation is optional.

Example:

The initial content of the entry on **<destinationDetails>** list for the destination field **TAX_TYPE** with value IND (selected from the Lookup **TAXTYPE**):

```
<destinationDetails>
  <sequence>
    <fieldName>TAX_TYPE</fieldName>
    <fieldValue>IND</fieldValue>
  </destinationDetails>
```

With translation value **XXX** for **TAX_TYPE:IND**

```
<destinationDetails>
  <sequence>
    <fieldName>TAX_TYPE</fieldname>
    <fieldValue>XXX</fieldValue>
  </destinationDetails>
```

With blank translation value for **TAX_TYPE:IND**

```
<destinationDetails>
  <sequence>
    <fieldName>TAX_TYPE</fieldname>
    <fieldValue></fieldValue>
  </destinationDetails>
```

Without an entry in the DVM for **TAX_TYPE:IND**

```
<destinationDetails>
  <sequence>
    <fieldName>TAX_TYPE</fieldname>
    <fieldValue>IND</fieldValue>
  </destinationDetails>
```

OTSS_PaymentType

An entry is provided for values of the **TENDERTYPE** lookup included in the base product. The column **OTSS_PaymentTypeCode** contains the lookup value. Enter the corresponding value from your revenue management system in the **EXT_PaymentTypeCode** column.

Example:

The initial content of the node is **<paymentType>CHECKING</paymentType>**:

With translation value **XXX** for **CHECKING**

```
<paymentType>XXX</payDestinationType>
```

With blank translation value for **CHECKING**

```
<paymentType></paymentType>
```

Without an entry in the DVM for **CHECKING**

```
<paymentType>CHECKING</paymentType>
```

OPC_PaymentType

This DVM is used exclusively for the integration with Official Payment Corp. It is used to map account types provided by Official Payments Corp. The base product includes the entry for each Official Payments' Corp. 'account type' in the column **OPC_PaymentType**.

For each entry specify the corresponding translation values from your revenue management system in the column **EXT_PaymentType**

OTSS_MessageNumbers

Map the expected return message numbers from the revenue management system to the messages defined in self service portal application.

How to Enable a New Payment Destination

Design Considerations

Determine what information your revenue management system requires in order to process this payment. It is usually one or more identifiers of the payment target (aka, destination), such as tax bill number, parcel number, filing year and tax type, etc. Define what input data is required and what's optional.

Determine what identification details casual self service user supposed to provide for identity verification. This will define what Identification Request you will be using. One identification request **IDENTIFY_TAX_PAYMENT** is provided with the base product.

You may also decide that a stand-alone identification step is not required, and incorporate taxpayer identification details into payment destination fields' collection.

Configuration

Configure Payment Destination. Specify Identification Request. Use existing or create new Fields. Define field's display order; create the footer help (HTML or text). In SOA/BPEL layer add the mapping for payment destination code, specific field's values and fee requirements for external payment providers (if applicable). Also add the mapping for the response message

Implementation

The one-time payment web service implementation in the revenue management system has to support the additional payment destination. Refer to that product's documentation for details.

How to Enable a New Payment Provider

Design Considerations

Determine whether the new payment services provider accepts payment requests via HTTP Post.

Determine what information this provider expects to receive with HTTP/S Form Post.

- The web self service application may supply:
 - Payment Destination Code and Details (Sequence/Field Name/Field Value Collection)
 - Payment Amount
 - Unique ID to identify the request in both systems. By default it is generated by the SOA/BPEL composite.
- The revenue management system may supply additional information about the taxpayer, such as mailing address or business name and/or any other details requires by the provider. It may also determine and return the convenience fee indicator.
- The external payment details are collected from the revenue management system using a web service Prepare Payment Data. You can utilize the extension points in the SOA/BPEL composite that processing this web service to reach other sources of information, if needed.

Determine whether the external payment details received from the revenue management system should be further manipulated in order to fit provider's requirements. If so, you will have to implement Prepare External Data plug-in.

Determine what payment methods this provider supports (e.g., credit card, bank draft, etc.) and decide what payment you will delegate to this provider.

Configuration

Configure new Payment Provider. Specify the description that will appear on available provider's list as a hyperlink. Enter the help text that will appear on the available provider's list below the hyperlink. Use help to convey important details about the provider and the services available on the external web site. Specify the redirect URL (usually, provider's web site). Finally, specify the Prepare Payment Data plug-in you've implemented. Indicate payment methods supported by this provider.

Implementation

Implement Prepare Data plug-in if necessary. You can use it to massage the data received from Prepare Payment Data web service, interpret the convenience fee indicator and perform any other data manipulation needed to prepare the form data for HTTP Post.

Implement the handling of the post-back response from the provider. There is no universal recipe for the post-back processing, but some elements of the post-back message are likely to be common. It includes a transaction reference ID (e.g., credit card authorization code) that uniquely identifies the payment in provider's system and can be used to trace the payment further to the credit card company or the bank. It is also likely to contain the same unique interaction (session) ID that was send with the HTTP Post.

Integration with Official Payments Corporation is provided with the base product. It includes the SOA/BPEL process that interprets the post-back XML message and transforms it into a One-Time Payment web service request. It also includes the processing of a payment report file by another SOA/BPEL process and the web service Payment Report Request that is generic enough and can be re-used.

FAQ: Payment

This section includes some frequently asked questions.

- **What if the Collection Notice in my system is represented differently; e.g., it contains two identification numbers instead of one?**

Configure a new payment destination according to your requirements. Create new Fields that represent your collection notice identification details.

- **According to my requirements, taxpayers have to provide extra POI details for certain payment options. How do I fulfill this requirement?**

Setup a Taxpayer Identification Service Request according to your requirements. Then attach this request to the appropriate payment destination.

- **There is no such thing as Collection Notice in my system. It is called Payment Voucher for Collection Note. Do I have to create a new payment destination?**

No, you don't. If the format of the field COLLECTION_NOTICE_NO is satisfying your requirements, you can override the description of the payment destination and set the display description of destination's field. You can also override the destination field's help text (the text that appears when you hover with the cursor over the input field on the screen).

- **In our system, the Collection Notice Number should be entered in a specific format, e.g., 999-999999-XX. Can I modify the provided payment destination configuration so it will enforce this rule?**

Yes, it is possible to apply a validation to the input value as long as this validation can be implemented using the standard regular expression syntax. Create a new Validation Rule and write a regular expression to represent your format restrictions. Then associate this new Validation Rule with payment destination's field.

- According to our business rules, users aren't allowed to make collection notice payments online. How can I disable this option?

Simply de-activate this payment destination.

- I want to start supporting a liquor license fee payments. How do I enable a new payment option?

Please refer to [How to Enable a New Payment Destination](#).

Chapter 6

Track Your Transaction

Overview

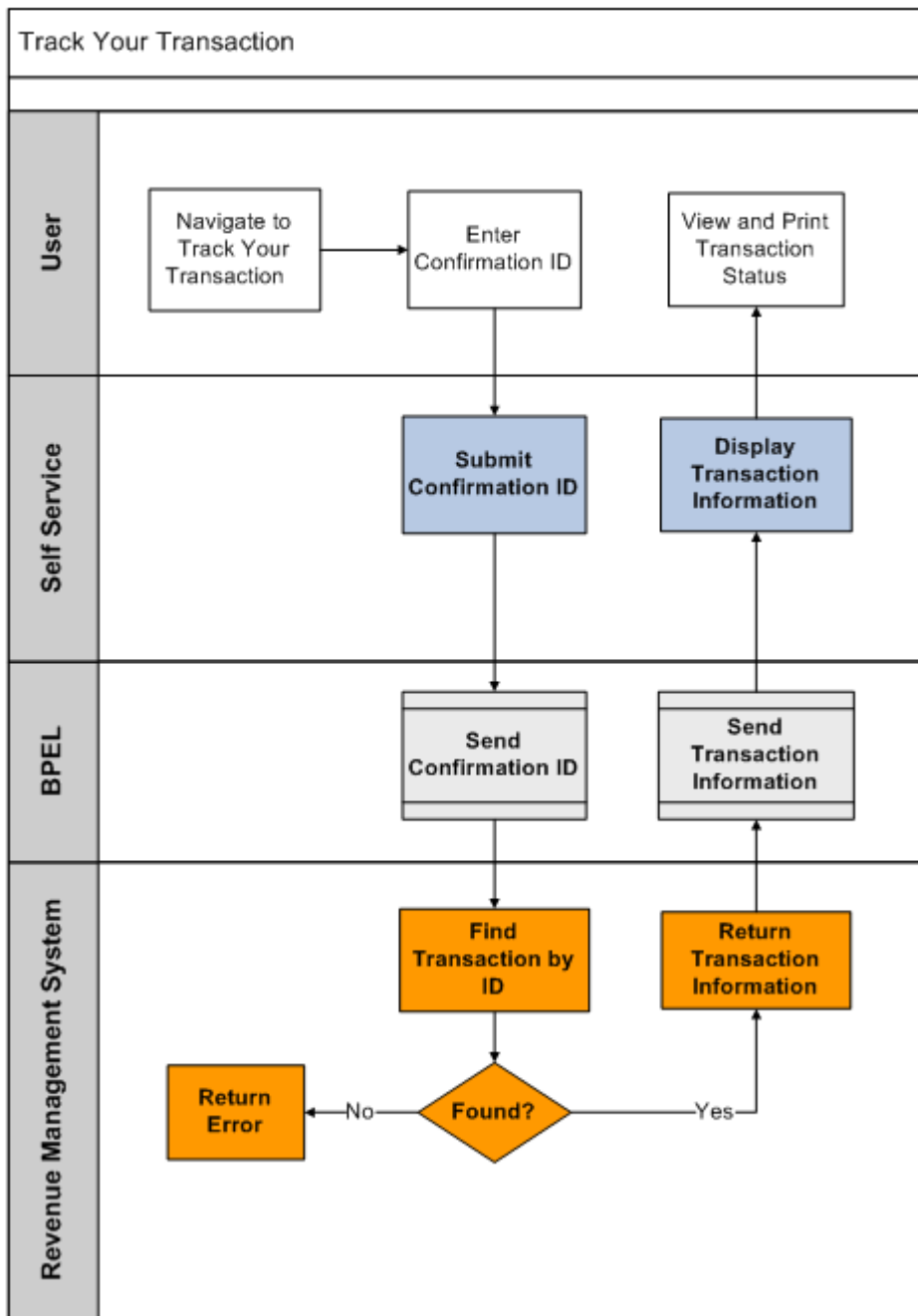
When a self-service user performs an operation that requires some action in the revenue management system he or she receives an acknowledgment that the request went through successfully. This acknowledgement is called a "confirmation". It contains a unique ID and a message with the current state of the request. The confirmation ID is recorded in the revenue management system, in the transaction created for the user request, in order to track the user's request and verify its status anytime from initiation to completion.

The self-service Track Your Transaction feature provides the user with the ability to trace the request using the confirmation number to trace his request using Track Your Transaction feature.

The **Track Your Transaction** option is accessible from the **On-Line Services** pull-down menu on the web self service portal navigation bar and from the left-side navigation panel.

Track Your Transaction

The following diagram describes the process flow for the Track Your Transaction feature.



A user navigates to the Track Your Transaction UI and enters the Confirmation ID for the transaction.

The **TSGetConfirmationInformation** web service is triggered and the collected input is sent to the revenue management system. The successful response contains a message that reflects the transaction status.

Track Your Transaction BPEL Processes

The Track Your Transaction request follows the BPEL *Synchronous Flow Without Confirmation ID*.

BPEL DVM Mapping

OTSS_MessageNumber

This DVM contains a cross-reference between messages defined in the self-service application and the revenue management system. It maps the revenue management **System** message category and message number combination to a self-service message number. The message category and message number are stored using a separator.

Add an entry to this table for every message defined in the application.

Implementing the Track Your Transaction Query

This section provides information required to fully implement the Track Your Transaction Query provided with the product.

Also see the "Confirmation Inquiry WSDL Node" in [Appendix A](#) for additional information, including a description of the WSDL nodes relevant to the Track Your Transaction feature, as well as the **TSGetConfirmationInformation** schema.

Web Services

To support self-service payments, the revenue management system is expected to implement the following web service:

Web Service	Description
TSGetConfirmationInformation	This Web Service accepts Confirmation ID as input and returns confirmation information.

See "Confirmation Inquiry WSDL Node" in [Appendix A](#) for additional information, including a description of the WSDL nodes relevant to the Track Your Transaction feature, and a description of the **TSGetConfirmationInformation** schema.

Important: The response is expected to contain both the **message category** (alphanumeric) and the **message number** for both error and confirmation messages in order to be translated properly in the BPEL Domain Value Mappings (aka, DVMs). If your revenue management system does not use a message category attribute, populate the `<messageCategory>` element with a dummy number, e.g., **1**.

UI Customization

The Track Your Transaction UI can be customized as follows:

- Using WebCenter Composer, insert html/text content at the top and/or the bottom of the page. These content items provide the guidelines on querying the status of a transaction by Confirmation ID. If the taxpayer is required to perform extra steps before searching, those steps can be outlined here.

Note: This can be useful when the customer has multiple back-end applications and each of these applications produces its own confirmation. A user may, for example, need to add a back-end system-specific "code" to the confirmation number before issuing the search (as is the case with the external payment provider Official Payments, which requires that an OPC code be added). For more information, see [Adding Content to a Portal Page](#) and [Customizing Help Content](#).

- Online help can also be customized to offer information or instructions in a specific context.

Chapter 7

Interactive Tax Assistant

Interactive Tax Assistant Overview

The Interactive Tax Assistant (ITA) feature allows the revenue authority to provide an interactive aid to the taxpayer. It helps with common questions regarding filing, credits, deductions, loans, and other tax policies.

ITA guides the taxpayer through an interview and provides answers based on the taxpayers's input. This feature is implemented using the Oracle Policy Automation(OPA) solution.

Web Interviews

Interactive interviews assist the taxpayer to make educated tax law-related decisions. The OPA determination engine converts the policies into a series of a problem solving dialogs.

The process of creating these interviews starts from understanding which decisions can be automated and which polices are the good candidates for ITA. They typically fall into the following categories:

- **Eligibility Questions.** Answers to questions that require extensive knowledge of the legislation, cases, and policies, and that typically require assistance from a tax professional (questions such as "Am I eligible for a specific benefit? Can I apply for a credit or a deduction?"). To answer questions such as these, the policy should be translated into eligibility rules. For example, in order to be eligible for a specific credit, the taxpayer should satisfy certain criteria.
- **Calculators.** Tax policies and instructions often include tables with rules for how different credits, deductions, and withholdings should be calculated. These calculations are also condition-driven, and provide different paths for different types of taxpayers.

These are the types of decision flows which OPA can automate consistently and accurately. A policy expert or business analyst can write rules to be applied by the Determinations Engine. A taxpayer can then access the interview via the Interactive Tax Assistant page and answer relevant questions which will be used to generate advice or assistance.

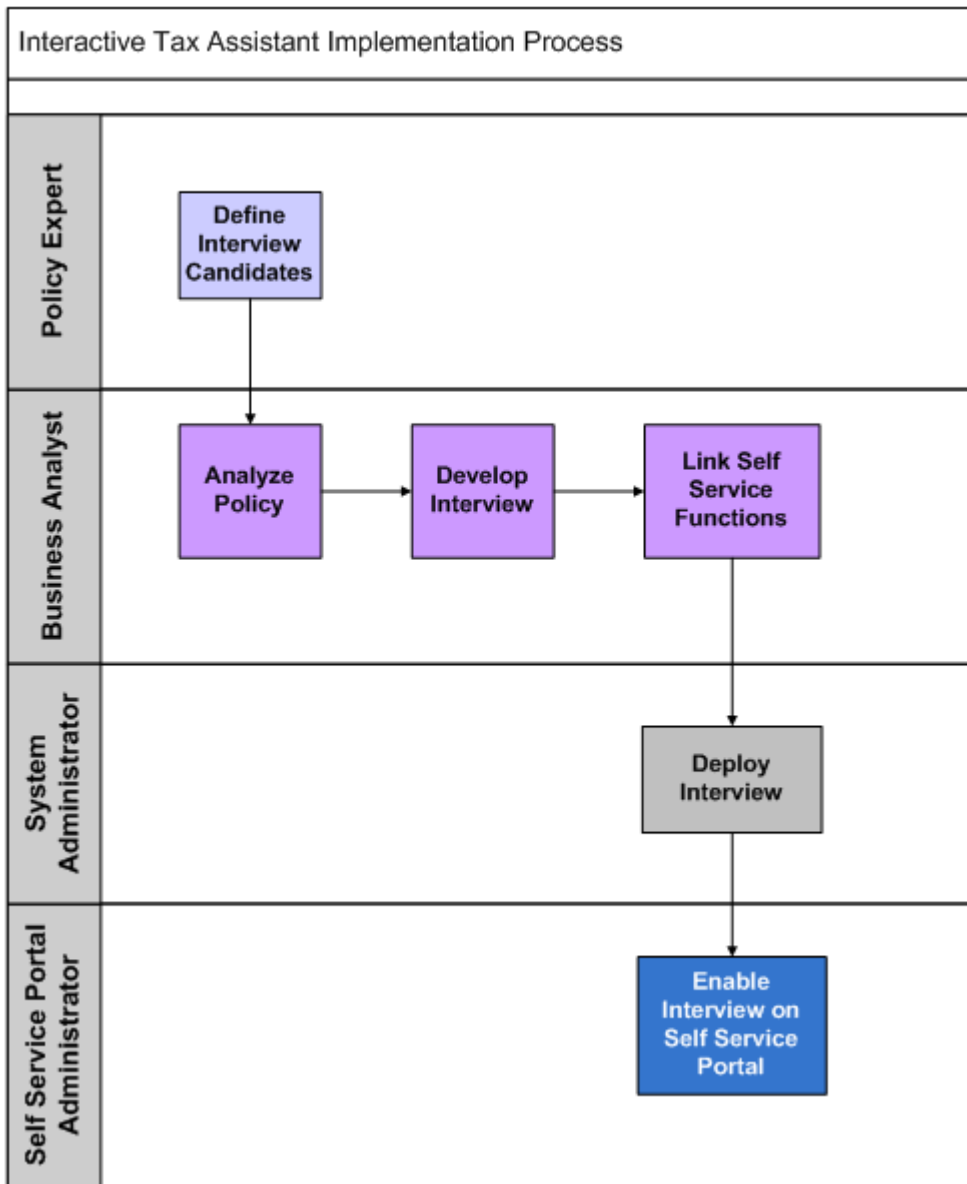
After web interviews are developed, tested, and deployed on a production server, they become available to taxpayers on the ITA page.

The site administrator is responsible for defining a new Interview Set and/or adding a new interview to the Interview Set.

Note: Oracle Policy Automation is a standalone product that has its own documentation and training manuals. They should receive adequate OPA training in order to learn how to write Web interviews. This document provides only supplemental information about OPA rules development.

Implementing Interactive Tax Assistant

Process Flow



This diagram illustrates the process of implementing a new Interactive Tax Assistant.

The roles and tasks involved in the process are as follows:

Policy Expert

Task: Define Interview Candidates

During this phase, the revenue agency policy expert analyzes legislation and defines candidates for the web interview. Call center statistics can be used to determine which policies are subject to the highest number of calls. These policies are then recommended for inclusion in the list of policies that should be automated.

Business Analyst

Task: Analyze Policy

The business analyst works with the content of the legislation to model the OPA rule. He or she determines what parts of the legislation should be modeled, and creates a design document that reflects the core structure of the model. The design document is validated by a policy expert to verify scope and design assumptions.

Task: Develop Interview

The business analyst transforms the rule design into the actual rule, implementing the logic, designing the screens, and creating help files. When development is complete, the rule is verified by a policy expert for the proper interpretation.

Task: Link Self-Service Application Functions

In some cases, an interview determination can trigger different self-service application actions. For example, if an interview determines which tax form a taxpayer should use, the taxpayer may be presented with the form so it can be filed right away. If a rule determines that the taxpayer is eligible for a specific program, the taxpayer may be presented with a hyperlink to the appropriate application form from the interview summary screen.

System Administrator

Task: Deploy Interview

When the rule is finalized, the system administrator deploys it on the server.

Self-Service Application Administrator

Task: Enable Interview on the Portal

The administrator uses the Interactive Tax Assistant portal to:

- Set up a new interview set
- Specify the location of the rules
- Select the rules that should be enabled

Defining Interview Sets

An Interview Set defines the logical group of the interactive interviews residing on the same Web Determinations Server.

Interview Set Search

Users can search for an existing Interview Set.

The standard **Edit** and **Delete** functions are available for each Interview Set.

You can click the **Add** button to define a new Interview Set.

Interview Set Maintenance

Interview Set - Main

Interview Set is a user-defined code that uniquely identifies the set.

Active indicates whether this Interview Set is ready for use. Only active sets are included on the list of Interview Sets displayed on the Tax Assistant page.

Description is the text displayed as the name of the top tree node on the Tax Assistant page.

Web Determinations Server represents the location of the Web Determinations Server (as a URL) where the interviews from this set are deployed.

Customizable indicates whether an implementation can modify the interviews list.

Detailed Description is for internal use.

Appearance

Help information can be entered as HTML or a plain text.

Override Help can be used to replace the Help provided with the base product

Interview Set - Interviews

This tab displays the list of Interviews.

When the new Interview Set is being added, the interview list is empty. Click the Load Interviews button in order to retrieve the full list of the interviews deployed on the Web Determinations Server referenced by the set.

The standard Edit and Delete functions are available for each field on the list.

Interview

Interview Name uniquely identifies the interview. It is derived automatically from the server.

Description appears as the tree node text (hyperlink) on the Tax Assistant page.

Display Sequence controls the order of interviews on the screen.

Launch Method indicates whether the interview should be launched in a separate popup window or be displayed on the main page area.

OPA Configuration and Customization

The interviews are embedded in the web self service portal page using the methods recommended by the official OPA documentation. The following configurations are implemented in order to achieve a look & feel that matches the host self-service portal page:

- The interview progress bar, status bar and several menu options are disabled in appearance properties.
- The interview screen header and footer areas are removed from the appropriate velocity templates.
- Fonts, colors and styles are adjusted in main.css.

Advanced Navigation

An Interview can be invoked directly from the HTML content anywhere on the website, using the following syntax:

```
/faces/oracle/webcenter/portalapp/pages/IntgTaxAssist.jspx?  
interviewSetCd=<Interview Set Code>&interviewCd=<Interview Name>
```

Chapter 8

SOA/BPEL Integration

Integration Overview

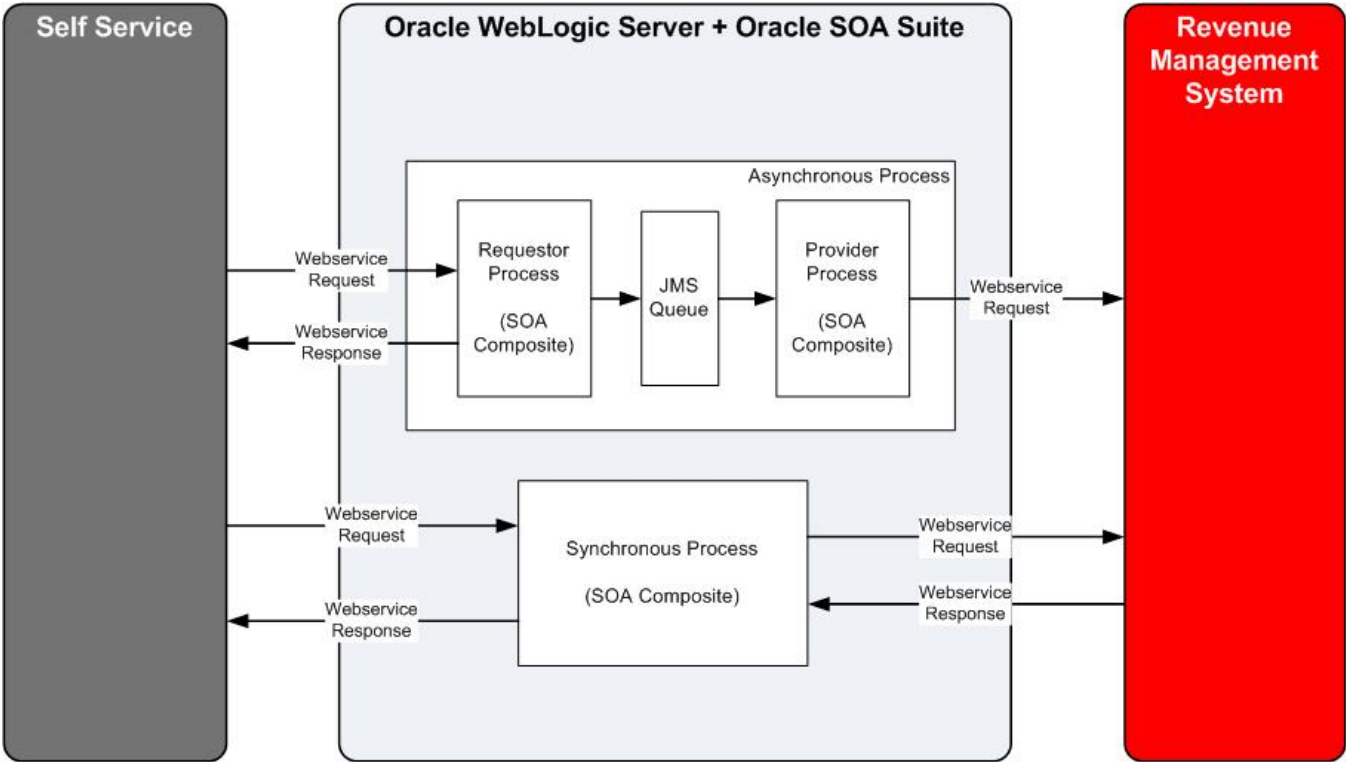


Figure 1: Integration Process Flows

This integration is an AIA Direct Integration.

All interaction between the self-service application, the integration layer, and the revenue management system use web services.

The integration layer is developed using SOA Suite installed on a WebLogic 11g application server.

The self-service application invokes web services hosted in the integration layer.

The integration layer comprises synchronous and asynchronous SOA composites.

The asynchronous processes use JMS Queues as a means of guaranteeing message delivery.

The integration SOA composites invoke the revenue management system web services.

The integration layer sends a response back to the self-service application.

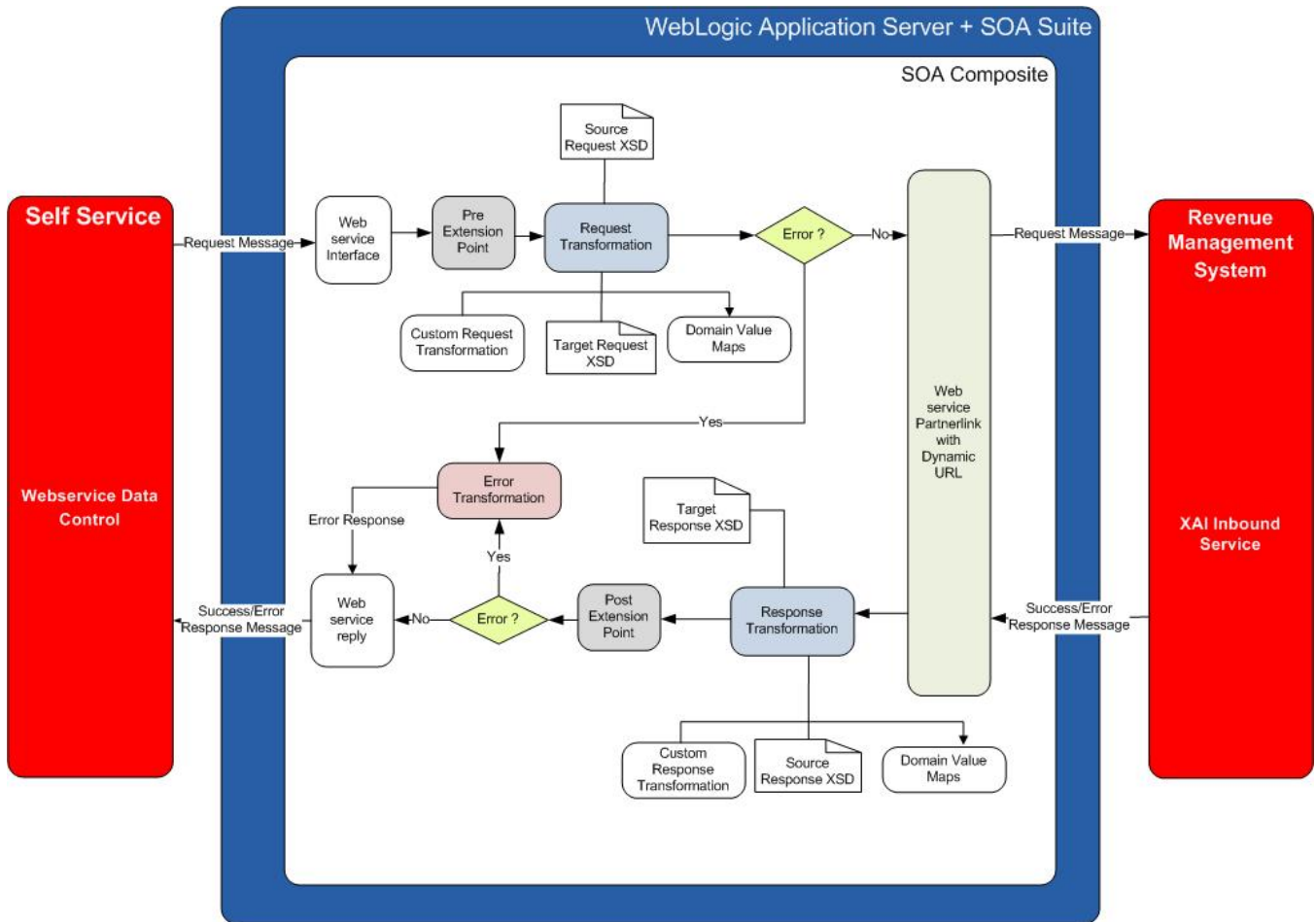
Integration Flow Patterns

Integration flows between the self-service application and the revenue management system are designed according to the following patterns:

- Synchronous flow without confirmation ID.
- Synchronous flow with confirmation ID.
- Asynchronous flow with confirmation ID.

The integration between the self-service application and Official Payments Corporation (external payment services) includes two special asynchronous flows.

Synchronous Flow Without Confirmation ID

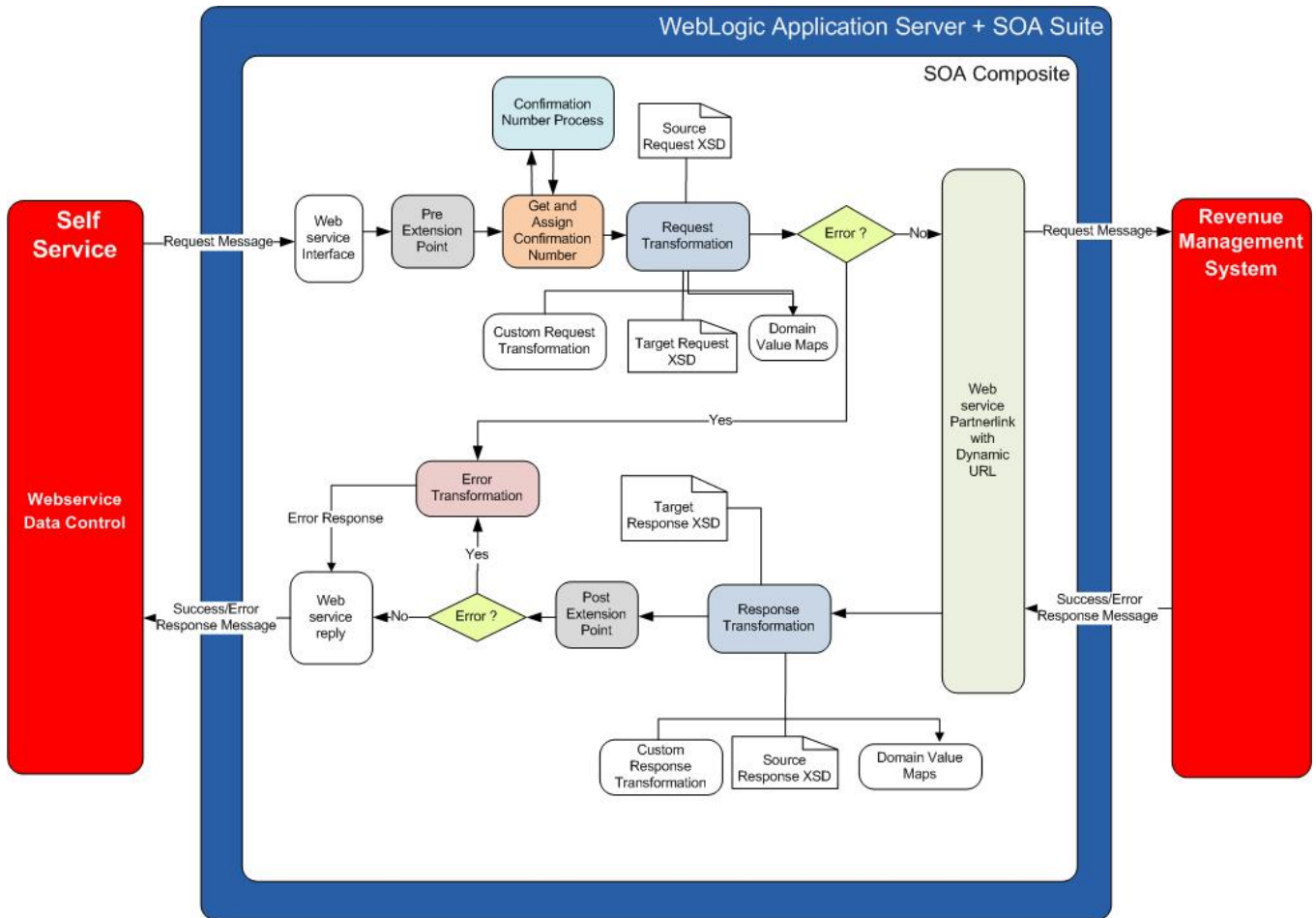


- The self-service application initiates the flow and sends a request (web service) to the integration layer.
- The integration SOA composite gets the XML message and transforms it from the self-service system format into the target system format.
- The SOA composite then forwards the request (web service) to the target system and receives the response.
- The response is transformed and forwarded to the self-service application.
- The SOA composites use DVMs for admin data translation.
- The integration layer provides pre- and post-transformation extension points as well as an extension point for custom transformations.
- The target web service end-point URL is configurable (see [Setting System Properties](#) for more details).

Synchronous Flow With Transaction ID Staging

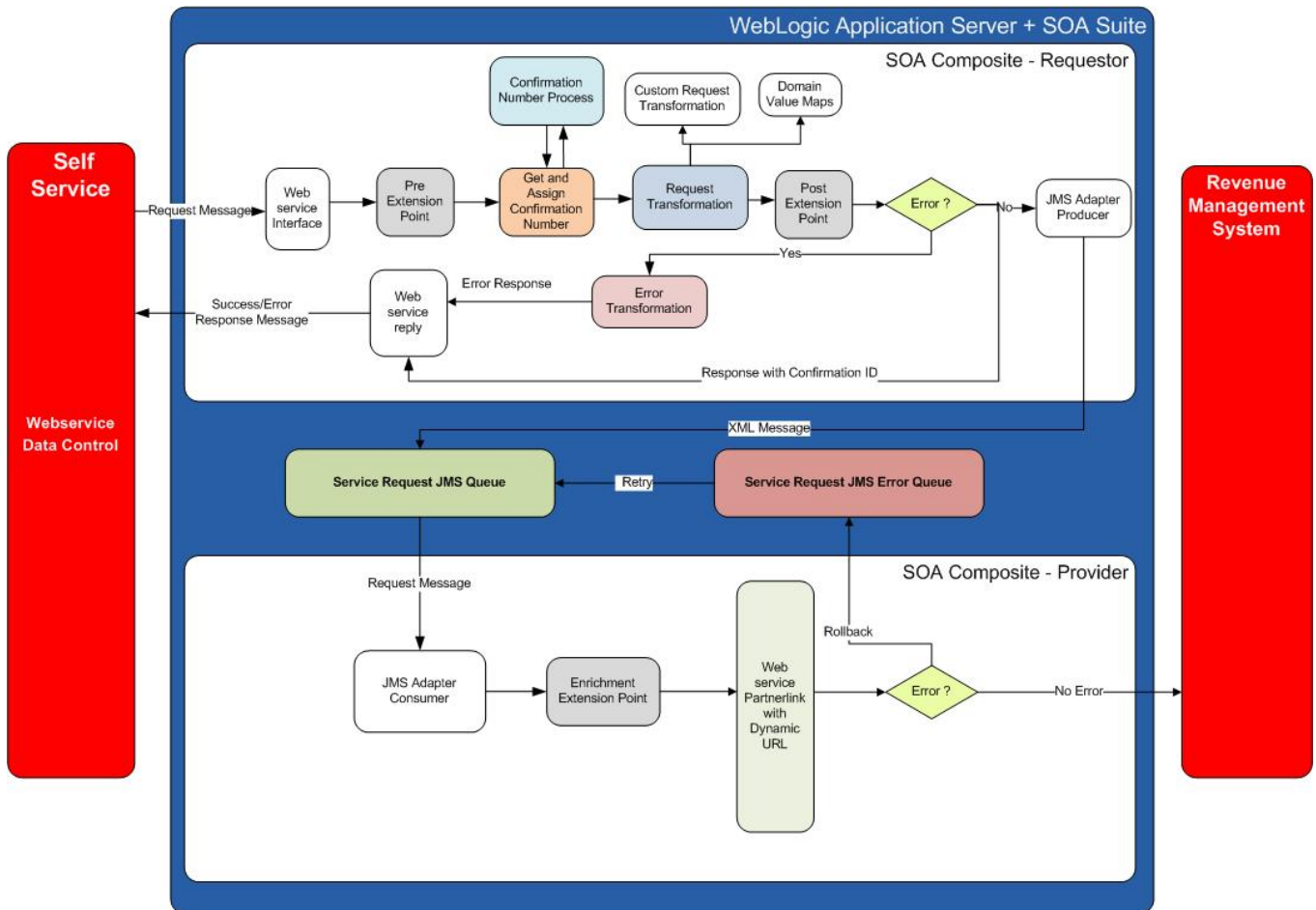
This flow includes an additional step during response transformation; it stores a record in the internal staging table. Otherwise, this pattern is the same as [Synchronous Flow Without Confirmation ID](#).

Synchronous Flow With Confirmation ID



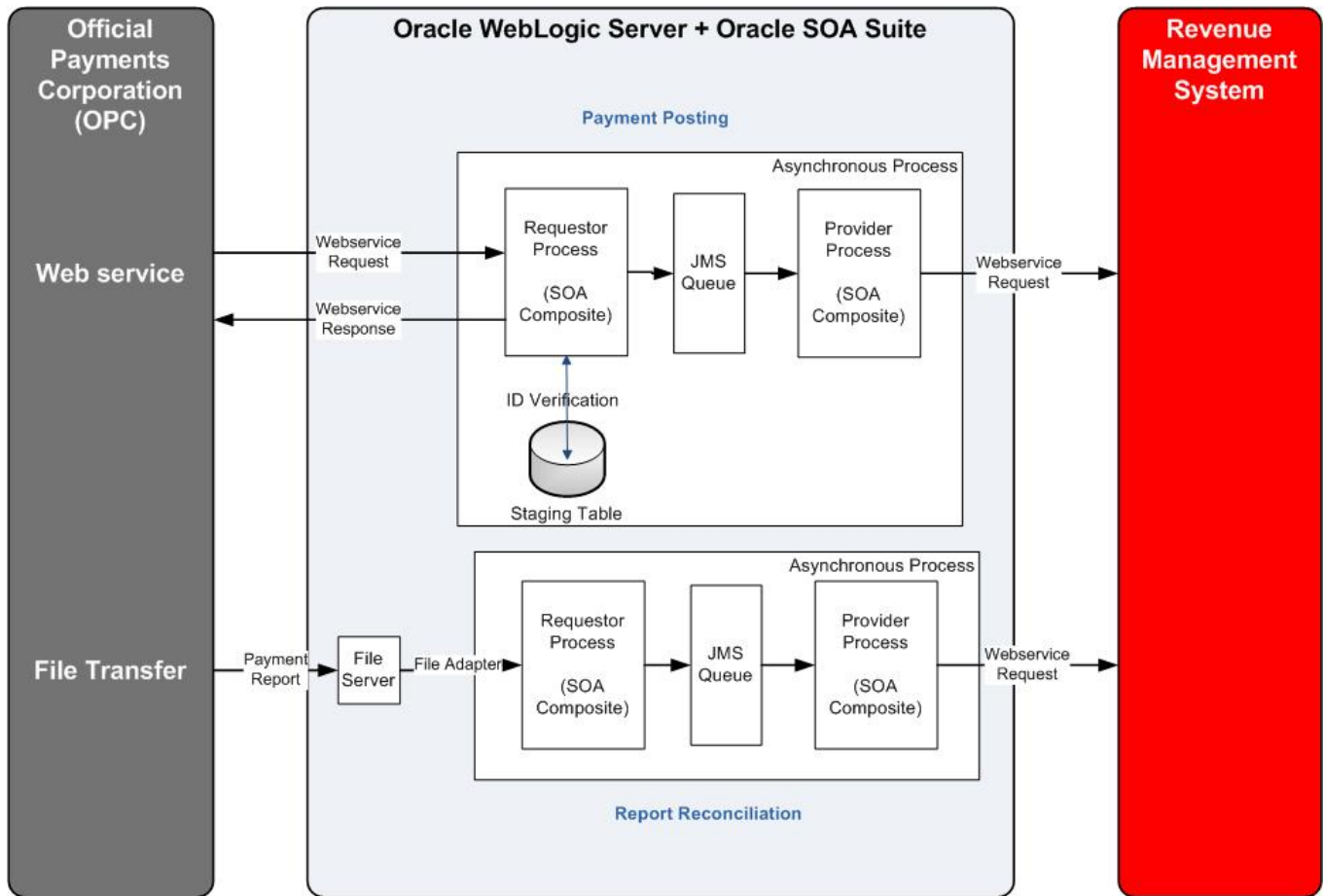
- The self-service application initiates the flow and sends a request (web service) to the integration layer.
- The integration SOA composite gets the XML message and transforms it from the self-service system format into target system format.
- The *Confirmation Number Utility Service* is invoked and a unique confirmation number is populated on the request XML <confirmationId> element. The default service is provided in the base product and SOA composites are initially configured to invoke it (see *Setting System Properties*). Implementation may create an alternative confirmation number generation service and reconfigure all or some of the integration flows to use an alternative.
- The SOA composite then forwards the request (web service) to the target system and receives the response.
- The response is transformed and forwarded to the self-service application.
- The SOA composites use DVMs for admin data translation.
- The integration layer provides pre- and post-transformation extension points, as well as an extension point for custom transformations.
- The target web service end-point URL is configurable (see *Setting System Properties* for more details).

Asynchronous Flow With Confirmation ID



- This flow is asynchronous.
- The self-service application expects an immediate response with a confirmation ID from the integration layer.
- The integration layer invokes the *Confirmation Number Utility Service* and generates the confirmation ID.
- The request message is transformed into the target system format and the confirmation number is populated on the request XML <confirmationId> element.
- The integration layer includes two independent SOA composites:
 - **Requestor** receives the request message from the self-service application, transforms it, adds it to a JMS queue and sends a response stamped with the confirmation number back to the self-service application.
 - **Provider** reads the message from the JMS queue and invokes the web service call to the target system. It does not wait for the response. If the provider is unable to invoke the web service, the message is rolled back to the Error JMS queue.
- The message flow from the self-service application to the Requestor is synchronous.
- The JMS queue breaks the transaction and the transaction from Provider to the target system is again synchronous.
- On delivery failure the messages is rolled back by the Provider to the Error queue and can be retried (see retry instructions below).

Flows for Official Payments Corporation Integration



This integration includes two flows created for the interaction with payment services provider Official Payments Corporation.

Both flows are asynchronous; no response of any kind is expected.

The Payment Posting process includes:

- Transaction verification against the record stored in the staging table.
- The special logic for confirmation number creation.

Integration Points

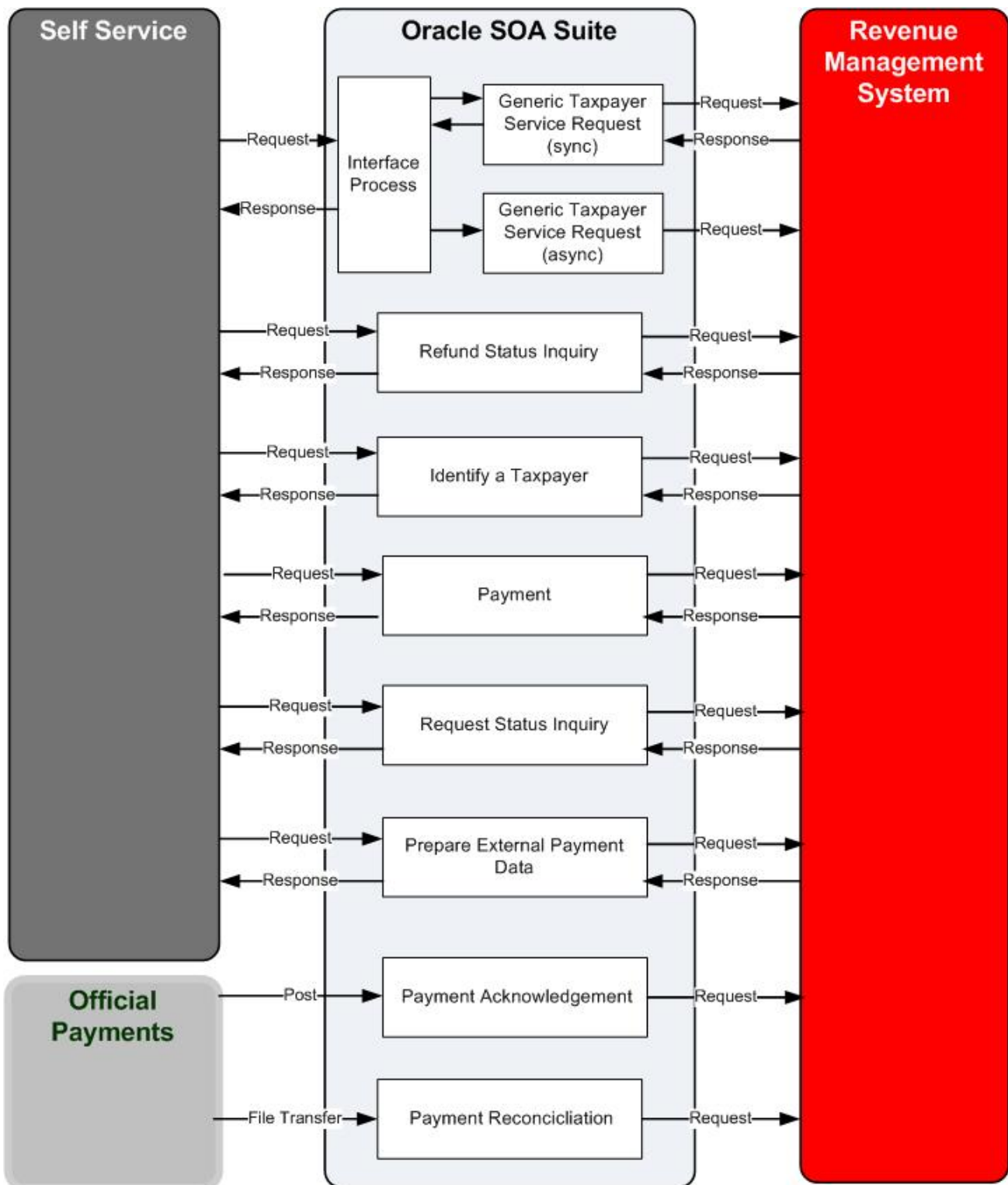


Figure 2: Self-Service – revenue management Integration Points Diagram

Flow Name	Description	Integration Flow Pattern
Generic Taxpayer Service Request	OTSS initiated	Combination of sync and async process with limited response (confirmation id)
Refund Status Inquiry	OTSS initiated	Sync flow with confirmation id
Identify a Taxpayer	OTSS initiated	Sync flow without confirmation id
Payment	OTSS initiated	Sync flow with confirmation id
Request Status Inquiry	OTSS initiated	Sync flow without confirmation id
Prepare External Payment Data	OTSS initiated	Sync flow without confirmation id
Payment Posting	OPC initiated	Async flow with no response
Report Reconciliation	OPC initiated	Async flow with no response

Integration Solution Flows

Flow Name	Description	Integration Flow Pattern
Generic Taxpayer Service Request	Self-service application initiated	Combination of sync and async process with confirmation id
Refund Status Inquiry	Self-service application initiated	Sync flow with confirmation id
Identify a Taxpayer	Self-service application initiated	Sync flow without confirmation id
Payment	Self-service application initiated	Sync flow with confirmation id
Request Status Inquiry	Self-service application initiated	Sync flow without confirmation id
Prepare External Payment Data	Self-service application initiated	Sync flow without confirmation id
Payment Posting	OPC initiated	Async flow with no response
Report Reconciliation	OPC initiated	Async flow with no response

Common Features of All BPEL Processes

- BPEL has pre-transformation extension point and post-transformation extension points as explained in the "Integration Extensibility" section of this document.
- BPEL processes have extension templates for each level of the outgoing message xml schema as explained in section Integration Extensibility.
- Optional Email notifications are sent out for business and technical failures.
- All endpoint invocation use dynamic partnerlink URL and must be configurable.
- For confirmation number the Confirmation number utility service is called using dynamic partnerlink.
- All processes have pre-transformation and post-transformation extension points as explained in the [Integration Extensibility](#) section.
- All processes have extension templates for each level of the outgoing message xml schema as explained in the [Integration Extensibility](#) section.
- Optional Email notifications will be sent out for the critical issues within selected asynchronous processes.
- All endpoint invocations use dynamic partnerlink URL. The URLs are configurable; see [Service Configurations](#) for details.
- For a confirmation number, the [Confirmation Number Utility Service](#) is called using dynamic partnerlink.

Common Mapping Rules

All self-service application messages share Audit, Custom Data, Confirmation, and Error xml fragments. The common mapping rules apply for all of these XML fragments.

Message Numbers, Message Parameters, and Currency

OTSS XML Fragment	ETPM XML Fragment
<pre><messageNumber/> <messageParameters> <parameters type="list"> <parameterType/> <parameterValue/> </parameters> </messageParameters> <currency/></pre>	<pre><messageCategory/> <messageNumber/> <messageParameters> <parameters type="list"> <parameterType/> <parameterValue/> </parameters> </messageParameters> <currency/></pre>

In the self-service application, Messages are keyed by number only. For this integration, the translation **OTSS_MessageNumber** DVM mapping is created to accommodate the combination of Message Category/Message Number. Specific rules apply when no mapping is found for the message number (for more, see the "Confirmation Message Error Message" and "Failed Message Delivery" topics).

The Message Parameter Type element is present on each entry of the Parameters collection. If supplied, it should be substituted using the corresponding value from the **OTSS_MessageParameterType** DVM mapping. If no mapping found, retain the value 'as is'.

Currency is required to support parameters whose type is **Amount**. The back-end application is expected to provide the appropriate value. If supplied, Currency codes should be substituted using **OTSS_Currency** DVM mapping. If no mapping is found, retain the value 'as is'.

Important: The response from the revenue management system is expected to contain both message category (alphanumeric) and message number for both error and confirmation messages in order to be translated properly in BPEL Domain Value Mappings (DVMs). If your revenue management system does not use a message category attribute, populate the <messageCategory> element with a dummy number (e.g., 1).

Confirmation Message (Fragment)

This common fragment will be included in multiple responses. It contains Confirmation Header group and Confirmation Details collection.

OTSS XML Fragment	ETPM XML Fragment
<pre><confirmation> <confirmHeader> <messageNumber/> <messageParameters> <parameters type="list"> <parameterType/> <parameterValue/> </parameters> </messageParameters> <currency/> <messageTxtOvr/></pre>	<pre><confirmation> <confirmHeader> <messageCategory/> <messageNumber/> <messageParameters> <parameters type="list"> <parameterType/> <parameterValue/> </parameters> </messageParameters> <currency/></pre>

OTSS XML Fragment

```

</confirmaHeader>
<confirmDetails type="list">
  <messageNumber/>
  <messageParameters>
    <parameters type="list">
      <parameterType/>
      <parameterValue/>
    </parameters>
  </messageParameters>
  <currency/>
  <messageTxtOvr/>
</confirmDetails>
</confirmation>

```

ETPM XML Fragment

```

  <messageTxtOvr/>
</confirmaHeader>
<confirmDetails type="list">
  <messageCategory/>
  <messageNumber/>
  <messageParameters>
    <parameters type="list">
      <parameterType/>
      <parameterValue/>
    </parameters>
  </messageParameters>
  <currency/>
  <messageTxtOvr/>
</confirmDetails>
</confirmation>

```

The default value in absence of DVM value setup for Message Number in the Confirmation Header is **0000**.

The default value in absence of DVM value setup for Message Number in the Confirmation Detail is **0004**.

In the absence of a DVM value setup for Message Number in the Confirmation Header, the replacement value is taken from the ConfigurationProperties.xml file (see [Module Configurations](#)). The value is initially set to **0000**.

In the absence of a DVM value setup for Message Number in the in the Confirmation Detail, the replacement value is taken from the ConfigurationProperties.xml (see [Module Configurations](#)). The value is initially set to **0004**.

Error Message (Fragment)

This common fragment is included in every response. It contains the error message group and status.

It is translated as follows:

- If the status indicator is set to **Error** but no mapping is found for the message number/category supplied, substitute it with the reserved generic error message code as defined in ConfigurationProperties.xml (see [Module Configurations](#)). The value is initially set to **0001**.
- If the status indicator is set to **Error** but no message number/category is supplied, no replacement occurs and the response XML is delivered 'as is'.

OTSS XML Fragment

```

<status/>
  <errorMsg>
    <messageNumber/>
    <messageParameters>
      <parameters type="list">
        <parameterType/>
        <parameterValue/>
      </parameters>
    </messageParameters>
    <currency/>
    <messageTxtOvr/>
  </errorMsg>

```

ETPM XML Fragment

```

  <status/>
  <errorMsg>
    <messageCategory/>
    <messageNumber/>
    <messageParameters>
      <parameters type="list">
        <parameterType/>
        <parameterValue/>
      </parameters>
    </messageParameters>
    <currency/>
    <messageTxtOvr/>
  </errorMsg>

```

Failed Message Delivery

Confirmation Id and details included in multiple messages. Back-end system response contains confirmation details or error message related to business rules.

However, confirmation ID is irrelevant if the error occurred during request transformation or message delivery failed to reach the back-end system (synchronous flow). If the status indicator is **Failed**, set Confirmation ID to blank, and populate the error message with the **0002** reserved number.

Under normal circumstances, the response contains confirmation details or a valid, business rule-based error message.

Special replacement rules apply if an error occurs during the request transformation or when message delivery fails to reach the target system (synchronous flow).

If the status indicator is **Failed**:

- The Confirmation ID is set to blank.
- The error message is populated with a message number defined in ConfigurationProperties.xml (see [Module Configurations](#)). The value is initially set to **0002**.

OTSS XML Fragment	ETPM XML Fragment
<pre><confirmationId/> ... <status/> <errorMsg> <messageNumber/> <messageParameters> <parameters type="list"> <parameterType/> <parameterValue/> </parameters> </messageParameters> <currency/> <messageTxtOvr/> </errorMsg></pre>	<pre><confirmationId/> ... <status/> <errorMsg> <messageCategory/> <messageNumber/> <messageParameters> <parameters type="list"> <parameterType/> <parameterValue/> </parameters> </messageParameters> <currency/> <messageTxtOvr/> </errorMsg></pre>

Domain Value Maps (DVM)

Domain value maps (DVMs) are a standard feature of the Oracle SOA Suite which maps codes and other static values across the applications. For example, country code **US** versus country code **USA**.

DVMs are static in nature, though Administrators can add additional maps as needed. Transactional business processes never update DVMs, they only read from them. DVMs are stored in XML files and cached in memory at runtime.

To maintain information within the domain value maps:

1. Open a browser and access the SOA Composer application (<http://host:port/soa/composer/>).
2. On the SOA Composer, click the **Open** dropdown and select **Open DVM**. This displays a list of all DVM files in MDS.
3. Select the relevant DVM you wish to maintain.
4. Edit the selected DVM. The **Edit** button in the top navigation bar enables editing the DVM.
5. Once the DVM has been edited, click **Save** in the navigation bar. This saves the DVM data for that session.
6. Click **Commit** after updating each DVM. This saves the DVM data in MDS.

The DVMs for this integration are listed in the following table:

Name	Integration Points	Purpose
OTSS_MessageNumber	All	<p>Maps revenue management message category/message number combination to self-service application message number. The message category and message number will be stored using a separator defined in ConfigurationProperties.xml</p> <p>Columns:</p> <ul style="list-style-type: none"> OTSS_MessageNumber EXT_MessageCategoryNumber
OTSS_MessageParameterType	All	<p>Maps revenue management message parameter type to the self-service application message parameter type.</p> <p>Columns:</p> <ul style="list-style-type: none"> OTSS_MessageParmType Supported delivered values: DATE, CURRENCY, STRING, NUMBER EXT_MessageParmType
OTSS_Currency	All	<p>Maps revenue management currency code to self-service application currency code.</p> <p>Columns:</p> <ul style="list-style-type: none"> OTSS_CurrencyCode Delivered default value: USD EXT_CurrencyCode
OTSS_ServiceRequestType	All service request-based integration points	<p>Maps revenue management service request type to self-service application service task type.</p> <p>Columns:</p> <ul style="list-style-type: none"> OTSS_SRTType Initially delivered values: supported service request types EXT_SRTType
OTSS_Field Codes	All	<p>This DVM is mainly for translating the values of the lookup-based fields. It maps self-service application field name/value combination to a corresponding value in revenue management system.</p> <p>Columns:</p> <ul style="list-style-type: none"> OTSS_FieldNameValue <p>Format:</p> <p>[FIELD NAME][separator][VALUE], where the separator character is defined in ConfigurationProperties.xml</p> <ul style="list-style-type: none"> EXT_FieldValue
OTSS_PaymentType	All payment-related integration points	<p>Maps revenue management payment [tender] type to self-service application payment type (values of the lookup TENDERTYPE).</p> <p>Columns:</p> <ul style="list-style-type: none"> OTSS_PaymentTypeCode Initially delivered values: CHECKING, SAVING, CREDIT EXT_PaymentTypeCode
OTSS_PaymentDestination	All payment-related integration points	<p>Maps revenue management payment destination to self-service application payment destination.</p> <p>Columns:</p> <ul style="list-style-type: none"> OTSS_PaymentDestValue Initially delivered values: supported payment destinations EXT_PaymentDestValue
OTSS_PaymentVendor	All payment-related integration points	<p>Maps revenue management payment vendor to self-service application payment provider.</p> <p>Columns:</p> <ul style="list-style-type: none"> OTSS_PaymentVendor EXT_PaymentVendor

Name	Integration Points	Purpose
OTSS_FeeRequirement	All payment-related integration points	<p>Maps a combination of revenue management payment vendor, payment destination, and fee requirement combination to self service application fee requirement.</p> <p>Columns:</p> <ul style="list-style-type: none"> OTSS_FeeRequirement EXT_FeeReq_PayVendor_PayDestination <p>Format:</p> <p>[FEE REQUIREMENT][separator]PAYMENT VENDOR][separator][PAYMENT DESTINATION]</p> <p>where the [separator] character is defined in ConfigurationProperties.xml</p>
OTSS_Country	All payment-related integration points	<p>Maps revenue management country codes to self-service and other applications country codes.</p> <p>Columns:</p> <ul style="list-style-type: none"> OTSS_Country EXT_Country
OPC_PaymentType	Official Payments Corp integration points	<p>Maps Official Payments Corp. payment[account]type code to revenue management payment [tender] type.</p> <p>Columns:</p> <ul style="list-style-type: none"> OPC_PaymentType <p>Valid values: V, VISA, M, MC, A, AMEX, D, DISC, ES, EC, EBS, EBC</p> <ul style="list-style-type: none"> EXT_PaymentType

Note:

For all processes/all DVMs except Message Numbers, the following rules apply:

- If the DVM mapping value is not found in the DVM data, the incoming value will be passed as is without any translation.
- If the mapping for the confirmation or error message number is not found, it is replaced with reserved generic confirmation or error message numbers. See [Common Mapping Rules](#) for more details.

Setting Configuration Properties

The ConfigurationProperties.XML file contains property values used by the integration components. Also, it contains various flags controlling the invocation of the extension points within the integration.

ConfigurationProperties.XML file is located in MDS under the directory `/apps/OTSS-ETPM/AIAMetaData/config`.

Note: Whenever the ConfigurationProperties.XML file is updated, it must be reloaded to MDS for updates to be reflected in the applications or services that use the updated properties. You can perform the reload by rebooting the SOA server.

Setting System Properties

Module Configurations are the properties that are shared by multiple integration flows within this integration.

Service Configurations are the properties that are used by a specific BPEL process.

Module Configurations

Configuration Property	Default Value	Description
SOA-INFRA.AuditLevel	ON	This property needs to be set to OFF if the Audit Level is set to off for the BPEL processes.
AdminEmailID	Administrator email id	This property will be used to set the administrator email id for the error handling process to send out an email in case of a critical failure where even the Error handling process fails.
ConfirmationHeader.Default.MessageNumber	0000	Default value for the message number in the Confirmation header.
Default.ErrorNumber	0001	Default Error number
Default.FailureNumber	0002	Default Failure number.
ConfirmationDetails.Default.MessageNumber	0004	Default value for message number in Confirmation Details section.
OTSS.SOA.DataSource		This is the database where the Payment Vendor Integration Reference table OTSS_PAYMENT_VENDOR_REF is located. In this table the integration flow OTSSGetExternalPaymentDataEBF stores external payment transaction staging records. The property is set during the installation check.
MessageCategoryNumber.Separator	:	
ExternalpaymentData.Status.Pending	PENDING	The status code for the new records in Payment Vendor Integration Reference (OTSS_PAYMENT_VENDOR_REF). Records in this status are created by OTSSGetExternalPaymentDataEBF .
ExternalpaymentData.Status.Processed	PROCESSED	OTSSPaymentPostingRequestEBF uses this status code for the processed records in Payment Vendor Integration Reference (OTSS_PAYMENT_VENDOR_REF) table.
ExternalpaymentData.ConfirmationID.Prefix	EVID	The prefix used for external payments.
OPC.ExternalId.EmailSubject	OPC ExternalID is Empty	The subject and the contents for the notification email sent when the unique transaction identifier is missing on the post-back XML (Official Payments Corp)
OPC.ExternalId.EmailContent	ExternalID from OPC is empty	
DB.ExternalId.EmailSubject	OTSS_PAYMENT_REF_ID is empty in OTSS_PAYMENT_VENDOR_REF	The subject and the contents for the notification email sent when there is no matching record found in the registration table for the unique transaction identifier from the post-back XML (Official Payments Corp)
DB.ExternalId.EmailContent	OTSS_PAYMENT_VENDOR_REF table does not have the following PaymentReferenceID/ExternalId .	
ResultCode.EmailSubject	ResultCode is not equal to A	The subject and the contents for the notification email sent when the post-back XML (Official Payments Corp) contains payment in error. In this scenario the post-back XML is not processed any further.
ResultCode.EmailContent	ResultCode value is not equal to A	
OPC.PaymentReport.Error.EmailSubject	Error while pushing the message to OTSSPaymentReport Queue	The subject and the contents for the notification email sent when the error occurs during payment report record processing (Official Payments Corp)
OPC.PaymentReport.Error.EmailContent	In the OTSSReportReconciliationRequestEBF while publishing message to the queue there is an error so rolled back the message to OTSSPaymentReport Error Queue	

Service Configurations

Configuration Property	Default/Delivered Value	Description
Service Name: OTSSIdentifyTaxpayerEBF		
Extension.PreXformOTSS	false	If set to true the request pre-transformation extension service will be invoked
Extension.PostXformOTSS	false	If set to true the request post-transformation extension service will be invoked
Extension.PreXformToOTSS	false	If set to true the response pre-transformation extension service will be invoked
Extension.PreXformToOTSS	false	If set to true the response post transformation extension service will be invoked
TaxpayerIdentification.Service.Name	{http://ouaf.oracle.com/spl/XAIXapp/xaiserver/TSTaxpayerIdentification}TSTaxpayerIdentificationService	
TaxpayerIdentification.Port.Name	TSTaxpayerIdentificationPort	
TaxpayerIdentification.Endpoint.URL	.../TSTaxpayerIdentification	Updated with the actual endpoint URL during the installation
FieldValue.Separator	:	
Service Name: OTSSRequestStatusInquiryEBF		
Extension.PreXformOTSS	false	If set to true the request pre-transformation extension service will be invoked
Extension.PostXformOTSS	false	If set to true the request post-transformation extension service will be invoked
Extension.PreXformToOTSS	false	If set to true the response pre-transformation extension service will be invoked
Extension.PreXformToOTSS	false	If set to true the response post transformation extension service will be invoked
ConfirmationInformation.Service.Name	{http://ouaf.oracle.com/spl/XAIXapp/xaiserver/TSGetConfirmationInformation}ConfirmationInformation Service	
ConfirmationInformation.Port.Name	TSGetConfirmationInformationPort	
ConfirmationInformation.Endpoint.URL	.../TSGetConfirmationInformation	Updated with the actual endpoint URL during the installation
RequestStatusInquiry.ConfirmationNumberService.Name	{http://xmlns.oracle.com/OTSS/OTSSConfirmationIdService/OTSSConfirmationIdBPPELProcess}otssconfirmationidbpelprocess_client_ep	
RequestStatusInquiry.ConfirmationNumberService.Port	OTSSConfirmationIdBPPELProcess_pt	

Configuration Property	Default/Delivered Value	Description
RequestStatusInquiry.ConfirmationNumberService.URL	.../soa-infra/services/OTSS-ETPM/OTSSConfirmationIdService/otssconfirmationidbpelprocess_client_ep	Updated with the actual endpoint URL during the installation
Service Name: OTSSPaymentEBF		
Extension.PreXformOTSS	false	If set to true the request pre-transformation extension service will be invoked
Extension.PostXformOTSS	false	If set to true the request post-transformation extension service will be invoked
Extension.PreXformToOTSS	false	If set to true the response pre-transformation extension service will be invoked
Extension.PreXformToOTSS	false	If set to true the response post transformation extension service will be invoked
PaymentSequence.Prefix	PTID	This is the prefix for the confirmation Ids generated for Payments
OneTimePayment.Service.Name	{http://ouaf.oracle.com/spl/XAIXapp/xaiserver/TSOneTimePayment}TSOneTimePaymentService	
OneTimePayment.Port.Name	TSOneTimePaymentPort	
OneTimePayment.Endpoint.URL	.../TSOneTimePayment	Updated with the actual endpoint URL during the installation
OneTimePayment.ConfirmationNumberService.Name	http://xmlns.oracle.com/OTSS/OTSSConfirmationIdService/OTSSConfirmationIdBPelProcess}otssconfirmationidbpelprocess_client_ep	
OneTimePayment.ConfirmationNumberService.Port	OTSSConfirmationIdBPelProcess_pt	
OneTimePayment.ConfirmationNumberService.URL	...soa-infra/services/OTSS-ETPM/OTSSConfirmationIdService/otssconfirmationidbpelprocess_client_ep	Updated with the actual URL during the installation
FieldValue.Separator	:	
Service Name: OTSSRefundStatusInquiryEBF		
Extension.PreXformOTSS	false	If set to true the request pre-transformation extension service will be invoked
Extension.PostXformOTSS	false	If set to true the request post-transformation extension service will be invoked
Extension.PreXformToOTSS	false	If set to true the response pre-transformation extension service will be invoked
Extension.PreXformToOTSS	false	If set to true the response post extension service will be invoked

Configuration Property	Default/Delivered Value	Description
		transformation extension service will be invoked
RefundStatusSequence.Prefix	RSID	This is the prefix for the confirmation Ids generated for Refund Status Inquiry requests
GetRefundStatusServiceName	{http://ouaf.oracle.com/spl/XAIXapp/xaiserver/TSGetRefundStatus}TSGetRefundStatusService	
GetRefundStatusPortType	TSGetRefundStatusPortType	
GetRefundStatusEndpoint	.../TSGetRefundStatus	Updated with the actual endpoint URL during the installation
RefundStatusInquiry.ConfirmationNumberService.Name	{http://xmlns.oracle.com/OTSS/OTSSConfirmationIdService/OTSSConfirmationIdBPELProcess}otssconfirmationidbpelprocess_client_ep	
RefundStatusInquiry.ConfirmationNumberService.Port	OTSSConfirmationIdBPELProcess_pt	
RefundStatusInquiry.ConfirmationNumberService.URL	...soa-infra/services/OTSS-ETPM/OTSSConfirmationIdService/otssconfirmationidbpelprocess_client_ep	Updated with the actual URL during the installation
FieldValue.Separator	:	
Service Name: OTSSTaxpayerServiceRequestProvider		
Extension.PreXform	false	If set to true the request pre-transformation extension service will be invoked
Extension.PostXform	false	If set to true the request post-transformation extension service will be invoked
ServiceRequest.Endpoint.URL	.../TSTaxpayerServiceRequest	Updated with the actual URL during the installation
Service Name: OTSSTaxpayerServiceRequestEBF		
Extension.PreXformOTSS	false	If set to true the request pre-transformation extension service will be invoked
Extension.PostXformOTSS	false	If set to true the request post-transformation extension service will be invoked
Extension.PreXformToOTSS	false	If set to true the response pre-transformation extension service will be invoked
Extension.PreXformToOTSS	false	If set to true the response post transformation extension service will be invoked
ServiceRequest.Endpoint.URL	.../TSTaxpayerServiceRequest	Updated with the actual URL during the installation

Configuration Property	Default/Delivered Value	Description
PaymentSequence.Prefix	SRID	This is the prefix for the confirmation Ids generated for Generic Service Requests
ServiceRequest.ConfirmationNumberService.Name	http://xmlns.oracle.com/OTSS/OTSSConfirmationIdService/OTSSConfirmationIdBPELProcess)otssconfirmationidbpelprocess_client_ep	
ServiceRequest.ConfirmationNumberService.Port	OTSSConfirmationIdBPELProcess_pt	
ServiceRequest.ConfirmationNumberService.URL	...soa-infra/services/OTSS-ETPM/OTSSConfirmationIdService/otssconfirmationidbpelprocess_client_ep	Updated with the actual endpoint URL during the installation
ServiceRequest.RequestMode.Sync	SYNCH	The values for the <responseMode> indicator on the request XML. Based on this indicator the message is processed following either asynchronous or synchronous flow.
ServiceRequest.RequestMode.Async	ASYNCH	
FieldValue.Separator	:	
Service Name: OTSSGetExternalPaymentDataEBF		
Extension.PreXformOTSS	false	If set to true the request pre-transformation extension service will be invoked
Extension.PostXformOTSS	false	If set to true the request post-transformation extension service will be invoked
Extension.PreXformToOTSS	false	If set to true the response pre-transformation extension service will be invoked
Extension.PreXformToOTSS	false	If set to true the response post transformation extension service will be invoked
ExternalPaymentData.Custom.Flag	false	If set to false , the default logic is triggered: a unique ID is generated, captured in the Payment Vendor Integration Reference (OTSS_PAYMENT_VENDOR_REF) table and populated in <externalId> node before the response is sent to the web self-service application. Otherwise, the custom extension service is invoked.
ExternalPaymentData.Service.Name	{http://ouaf.oracle.com/spl/XAIApp/xaiserver/TSPprepareExtPaymentData}TSPprepareExtPaymentDataService	

Configuration Property	Default/Delivered Value	Description
ExternalPaymentData.Port.Name	TSPPrepareExtPaymentDataPortType	
ExternalPaymentData.Endpoint.URL	.../TSPPrepareExtPaymentData	Updated with the actual URL during the installation
ExternalPaymentData.Action.Prepare	PREPARE	The values for the <action> on the request XML. When action is PREPARE , the process generates the unique ID and store is in Payment Vendor Integration Reference (OTSS_PAYMENT_VENDOR_REF) table before delivering the response to the web self-service application.
ExternalPaymentData.Action.Validate	VAIDATEONLY	
FeeRequirement.DVMValue.Separator	:	
FieldValue.Separator	:	
Service Name: OTSSPaymentPostingRequestEBF		
Extension.PreXformOTSS	false	If set to true the request pre-transformation extension service will be invoked
Extension.PostXformOTSS	false	If set to true the request post-transformation extension service will be invoked
TechnicalError.NotificationFlag	false	If set to true Technical error notification will be sent via email.
PaymentPostback.PaymentVendor		Currently not in use
PaymentPostback.CurrencyCode	USD	The default value supported by Official Payments Corporation
Service Name: OTSSPaymentPostingRequestProvider		
PaymentPostBack.Service.Name	{http://ouaf.oracle.com/spl/XAIXapp/xaiserver/TSTimePayment}TSTimePaymentService	
PaymentPostBack.Port.Name	TSTimePaymentPort	
PaymentPostBack.Endpoint.URL	.../TSTimePayment	Updated with the actual endpoint URL during the installation
Service Name: OTSSReportReconciliationRequestEBF		
Extension.PreXformOPC	false	If set to true the request pre-transformation extension service is invoked
Extension.PreXformTax	false	If set to true the request post-transformation extension service will be invoked

Configuration Property	Default/Delivered Value	Description
TechnicalError.NotificationFlag	false	Not in use
ReportReconciliation.PaymentVendor		This value has to be updated with the Payment Vendor code configured in the revenue management system for Official Payments Corporation
ReportReconciliation.Currency	USD	This is the default currency supported by Official Payments Corporation
Service name OTSSReportReconciliationRequestProvider		
BusinessError.NotificationFlag	false	If set to true Business error notification will be sent via email.
TechnicalError.NotificationFlag	false	If set to true Technical error notification will be sent via email.
Report.Endpoint.URL	.../TSPProcessExtPayReportRecord	Updated with the actual URL during the installation
Extension.PreXform	false	

Important: The **TechnicalError.NotificationFlag** and **BusinessError.NotificationFlag** properties found in the ConfigurationProperties.xml file are not currently in use. Do not remove them from the file; they are there for the future use.

Integration With the Revenue Management System

This section describes the integration flows participating in the integration between the self-service application and the revenue management system.

Confirmation Number Utility Service

Multiple integration processes populate a unique confirmation ID on the request XML message. This confirmation ID is delivered to the target system and can be used for transaction tracking purposes.

The Confirmation Number Utility Service encapsulates the common confirmation number generation logic. The integration flows which require confirmation number invoke this service using dynamic partnerlink URL.

The input to this service is a database sequence name; the output is the alphanumeric confirmation ID, formatted as follows:

- The next available sequence number is formatted as 12-digit number, left-padded with zeroes.
- The prefix is defined in the [Service Configurations](#) for this flow.

The delivered integration configuration includes a prefix for each flow.

The service uses a DB Adapter to get the next sequence number for the input sequence ID.

The sequence IDs and sequences are:

- **SRID** - ServiceRequestSequence
- **RSID** - RefundStatusSequence

- **PTID** – PaymentSequence

Also see Appendix B for a sample of the *GetConfirmationID Request/Response* messages.

Integration Services

Name	Description
OTSSConfirmationIdService	Confirmation number SOA composite. The process uses the synchronous flow which takes a sequence ID as input and returns a confirmation number as output.

Adapter Services

Name	Description
OTSSConfirmationIdAdapter	Confirmation number SOA composite. OTSS confirmation number DB adapter.

Payment Integration Flow

Business Details

The self-service application collects the payment details from the user and sends the information to the revenue management system. This request triggers the payment creation process. The self-service application receives a response: it contains either payment confirmation details and a confirmation ID, or error information.

Technical Details

This flow conforms to the *Synchronous Flow with Confirmation ID* pattern.

A self-service application issues a web service call to the integration layer and invokes the payment integration flow. The SOA composite process transforms the request message and performs DVM-based translations:

- **<payDestinationType>** - this element contains the Payment Destination code defined in the self-service application. It is translated using OTSS_PaymentDestination DVM.
- **<destinationDetails>** - this list contains field name/field value pairs and carry the details of the payment destination, "what the payment is for?". It may include dates, codes, identifiers, such as collection notice ID, or a tax type. The content of the **<fieldValue>** element may also belong to a lookup (predefined values list). The **<fieldValue>** node is translated using OTSS_Field Codes DVM, as follows:
 - A combination of **[field name][separator][field value]** from the self-service translated in to a **[value]** from the revenue management system. The separator is configured in *Service Configurations*.
- **<currency>** is translated using OTSS_Currency DVM.
- **<paymentType>** is translated using OTSS_PaymentType DVM. The value represents source of the payment: checking or savings bank account, credit card, or others.

The *Confirmation Number Utility Service* is called with input sequence name (specified in *Service Configurations* for OTSSPaymentEBF). The service returns a confirmation number and the process populates **<confirmationId>** element on the request message.

Once the transformation is completed, the request message is forwarded to the revenue management system via a web service call.

The response from the revenue management system is received by the integration layer.

The contents of the confirmation details or the error message are translated following the [Common Mapping Rules](#).

If integration encounters an exception (e.g., connectivity error, transformation error) while processing the message, integration will return a SOAP fault.

Integration Services

Name	Description
OTSSPaymentEBF	Payment enterprise business flow process. The process will use the synchronous flow with confirmation id. This process performs transformation using DVMs, and includes pre- and post-extension services, as well as custom transformations.

Web Services

Name	Description
TSSOneTimePayment	This web service is used to initiate the payment creation process in the revenue management system.

Prepare Payment Data Integration Flow

Business Details

This process is used in two modes:

- **Validation** - The self-service application collects payment destination details from the user and sends it to the revenue management system for verification. The verification logic checks the validity of the input (for example, whether the taxpayer is registered for the input tax type).
- **External Payment Details Preparation** – The self-service application sends the request to the revenue management system before redirecting the payment process to the external payment service. The request message contains payment destination details and payment amount collected from the user and also an external payment vendor’s identifier.

The response message contains the information that needs to be sent to the external payment service, for example, taxpayer’s mailing address or phone number. It also contains the fee requirement indicator.

The vendor usually charges an additional fee (convenience fee) for its services. The fee calculation is done by the vendor, but the revenue management system makes a determination whether the fee will be paid by the taxpayer or otherwise absorbed by the revenue authority.

When processing the response, the SOA composite also invokes the special logic that generates a unique ID for the interaction with external payment provider service and registers this transaction in the staging table.

Technical Details

This flow conforms to the [Synchronous Flow with Transaction ID Staging](#) pattern.

The self-service application sends the Prepare External Payment Data request in form of an XML message, which is transformed by the integration.

Upon successful transformation, the integration sends the request to the revenue management system (web service).

Depending on the value in the `<action>` element, the response is processed as follows:

- Action **VALIDATEONLY** – The response is transformed, the DVM-based translation is performed, and the message is forwarded back to the self-service application. The translation includes:

- Action **PREPARE** – If the response contains no errors, the process performs an optional transaction staging. The following is performed if **ExternalPaymentData.Custom.Flag** is set to **false** (see [Module Configurations](#)):
 - If the element **<externalId>** in the response message is empty, the integration generates a unique identifier for the anticipated interaction with the external payment service. The element **<externalId>** is populated with the newly-generated value. The new record is added to the Payment Vendor Integration Reference table OTSS_PAYMENT_VENDOR_REF:
 - Column OTSS_PAYMENT_REF_ID – A unique transaction ID.
 - Column OTSS_PAYMENT_VENDOR – Payment vendor identifier in the revenue management system.
 - Column OTSS_TAXPAYER_ID – The value of the **<taxpayerId>** element.
 - Column OTSS_BEHALF_OF_TAXPAYER_ID – The value of the **<onBehalfOfTaxpayerId>** element.
 - Column OTSS_PAYMENT_REF_STATUS – The value configured in the **ExternalPaymentData.Status.Pending** property (see [Module Configurations](#)).

The following DVM-based translations are performed for both response and request messages:

- **<paymentVendor>** is translated using OTSS_PaymentVendor DVM.
- **<payDestinationType>**, **<destinationDetails>**, **<currency>**, and **<paymentType>** are translated the same way they are translated by the [Payment Integration Flow](#).

The contents of the error message is translated following the [Common Mapping Rules](#).

If an exception (e.g., a connectivity or transformation error) is encountered while processing the message, the integration returns a SOAP fault.

Integration Services

Name	Description
OTSSGetExternalPaymentDataEBF	Prepare External Payment Data enterprise business flow process. The process uses the Synchronous Flow with Transaction ID Staging . This process includes transformation using DVMs, pre- and post-extension services, and custom transformations. In addition, this process generates a unique transaction ID and stages this ID in the Payment Vendor Integration Reference table. This step is optional, and the invocation is controlled by the ExternalPaymentData.Custom.Flag property.

Adapter Services

Name	Description
OTSSExternalPaymentDataDBAdapter	OTSS External Payment data DB Adapter. This DB adapter is used to write the unique transaction id to the Payment Vendor Integration Reference table.

Web Services

Name	Description
TSPrepareExtPaymentData	This web service is used to validate payment destination information provided by the taxpayer (action VALIDATEONLY) and to retrieve additional data for payments redirected to the external vendor (action is PREPARE).

Database Tables

The table **OTSS_PAYMENT_VENDOR_REF** stores the unique transaction reference ID for payments made through external payment services. It captures the transaction ID and the internal taxpayer IDs (from the revenue management system). A new pending record is added to this table each time the self-service application prepares to redirect the payment process to the external vendor. When the finalized payment posting message comes from the external vendor, the transaction can be verified using the transaction reference ID and the record is marked as processed.

Column Name	Datatype	Mapping
OTSS_PAYMENT_VENDOR	Varchar 50	paymentVendor
OTSS_PAYMENT_REF_ID	Varchar 50	externalId
OTSS_TAXPAYER_ID	Varchar 50	taxpayerId
OTSS_BEHALF_OF_TAXPAYER_ID	Varchar 50	onBehalfOfTaxpayerId
OTSS_PAYMENT_REF_STATUS	CHAR10	Integration populates the status specified in the ExternalpaymentData.Status.Pending property or ExternalpaymentData.Status.Processed property. For details on these properties, see Module Configurations .

Generic Taxpayer Request Integration Flow

Business Details

The self-service application collects the service request details from the taxpayer. The service request feature supports a wide range of business use cases and the handling of the specific service in the revenue management system, depending on the request type.

From the self-service user perspective, the service is either provided immediately, in which case the user receives confirmation details, or the request is accepted for future processing, in which case the user receives an acknowledgement message and a confirmation ID for the later inquiries.

The service request data is delivered to the revenue management system in the form of a generic name/value collection:

```
<serviceRequestData>
  <requestField type="list">
    <sequence>
      <fieldName>
        <fieldValue>
      </requestField>
    </serviceRequestData>
```

The request also includes an element containing the service request type code. The receiving service in the revenue management system orchestrates the request processing based on the service request type and triggers a corresponding business process. The response contains either the confirmation details or an error message.

Technical Details

The service request message can be delivered by either synchronous or asynchronous flows, both including confirmation ID generation. The service makes a determination based on the value of the **<responseMode>** element.

The integration invokes a different type of flow as follows:

- **SYNCH** – The *Synchronous Flow with Confirmation ID* is invoked. The request message is transformed, the values are translated using DVMs, the confirmation ID is generated and populated on the request, and the request is sent to

the revenue management system. The response message is transformed, translated, and returned to the self-service application.

- **ASYNCH** – The *Asynchronous Flow with Confirmation ID* is invoked. The confirmation ID is generated and populated on the message. The request message is added to the OTSSServiceRequest JMS Queue. The response with the confirmation ID and predefined generic confirmation message number is returned to the self-service application.

The provider process picks the message from the service request queue and invokes the revenue management system's web service.

The response from the revenue management system is received by the integration layer.

The process calls the *Confirmation Number Utility Service* with the input sequence name (specified in *Service Configurations* for OTSSTaxpayerServiceRequestEBF). The service returns a confirmation number and the process populates the <confirmationId> element on the request message.

DVM-based translations:

- <serviceRequestType> - This element contains the Service Request Type code defined in the self-service application. It is translated using OTSS_ServiceRequestType DVM.
- <requestField> - This list contains field name/field value pairs and carry the details of the service request. It may include dates, codes, identifiers, such as collection notice ID, or tax type. The content of the <fieldValue> element may also belong to a lookup (predefined values list). The <fieldValue> node is translated using OTSS_Field Codes DVM, as follows:
 - A combination of [field name][separator][field value] from the self-service application is translated to a [value] from the revenue management system. The separator is configured in *Service Configurations*.

The contents of the confirmation details or the error message are translated following the *Common Mapping Rules*.

If integration encounters an exception (e.g., a connectivity or transformation error) while processing the message, integration returns a SOAP fault.

Integration Services

Name	Description
OTSSTaxpayerServiceRequestEBF	Generic taxpayer service request enterprise business flow process. Based on requestMode, this process invokes the sync or async flow. This process will have transformation using DVMs, pre- and post-extension services, and custom transformations. For the async flow, a JMS adapter puts the message in the service request queue.
OTSSTaxpayerServiceRequestProvider	Generic taxpayer service request provider process. An SOA composite uses a JMS adapter to read the message from the service request queue and invokes the revenue management system web service. The service consists of an enrichment extension service and a dynamic endpoint URL. On error, the process will rollback the message to the error queue.

Adapter Services

Name	Description
OTSSServiceRequestJMSProducer	Self-service application service request JMS producer. JMS producer adapter to write to the self-service application service request queue.
OTSSServiceRequestJMSConsumer	OTSS service request JMS consumer. JMS consumer adapter to read from the OTSS service request queue.

JMS Queues

Name	Description
OTSSServiceRequest	Self-service application Service Request queue. Used by the integration layer to add transformed service request messages from the self-service application.
OTSSServiceRequestError	Self-service application Service Request error queue. Error Queue for self-service application Service Request.

Web Services

Name	Description
TSTaxpayerServiceRequest	This web service is used to process the input request details and check the status of the expected refund based on the information provided. The response contains refund status (confirmation) details and expected refund amounts.

Taxpayer Identification Integration Flow

Business Details

The self-service application collects the Proof Of Identity (POI) details from the taxpayer.

The information is verified in the revenue management system, and the response contains the list of Taxpayer IDs or an error message.

The request data is delivered to the revenue management system in the form of a generic name/value collection:

```
<serviceRequestData>
  <requestField type="list">
    <sequence>
      <fieldName>
      <fieldValue>
    </requestField>
  </serviceRequestData>
```

The request also includes an element containing a service request type code. The receiving service in the revenue management system orchestrates the request processing based on the service request type and triggers taxpayer identification processes.

Technical Details

This flow conforms to the *Synchronous Flow Without Confirmation ID* pattern.

The self-service application sends the information in form of an XML message. The message is transformed by the integration, translated using DVMs, and sent to the revenue management system. The response message is transformed, translated, and returned to the self-service application.

DVM-based translations:

- **<serviceRequestType>** - This element contains the Service Request Type code defined in the self-service application. It is translated using OTSS_ServiceRequestType DVM.
- **<requestField>** - This list contains field name/field value pairs and carry the details of the service request. It may include dates, codes, and identifiers, such as collection notice ID or tax type. The content of the **<fieldValue>** element

may also belong to a lookup (predefined values list). The `<fieldValue>` node is translated using OTSS_Field Codes DVM, as follows:

- A combination of `[field name][separator][field value]` from the self-service application is translated to a `[value]` from the revenue management system. The separator is configured in [Service Configurations](#).

The contents of the error message are translated following the [Common Mapping Rules](#).

If integration encounters an exception (e.g., a connectivity or transformation error) while processing the message, integration returns a SOAP fault.

Integration Services

Name	Description
OTSSIdentifyTaxpayerEBF	Identify a Taxpayer enterprise business flow process. The process will use the synchronous flow without confirmation ID. This process will have transformation using DVMs, pre- and post-extension services, and custom transformations.

Web Services

Name	Description
TSTaxpayerIdentification	This web service is used to trigger taxpayer identity verification in the revenue management system. The response includes a list of taxpayer IDs.

Refund Status Inquiry Integration Flow

Business Details

The self-service application collects the refund status inquiry details from the taxpayer.

The information is evaluated in the revenue management system, and the response contains the refund status details and related amounts, or an error message. The response message contains the confirmation ID that the taxpayer may use as a reference when contacting the revenue authority. The refund status information is returned in the confirmation details fragment.

The request data is delivered to the revenue management system in a form of a generic name/value collection:

```
<serviceRequestData>
  <requestField type="list">
    <sequence>
      <fieldName>
      <fieldValue>
    </requestField>
  </serviceRequestData>
```

The request also includes an element containing a service request type code. The receiving service in the revenue management system orchestrates the request processing based on the service request type and triggers taxpayer identification processes.

Technical Details

This flow conforms to the [Synchronous Flow with Confirmation ID](#) pattern.

The self-service application sends the Refund Status Inquiry information in the form of an XML message. The request message is transformed, the values are translated using DVMs, the confirmation ID is generated and populated on the

request, and the request is sent to the revenue management system. The response message is transformed, translated and returned to the self-service application.

The *Confirmation Number Utility Service* is called with input sequence name (specified in *Service Configurations* for OTSSRefundStatusInquiryEBF). The service returns a confirmation number and the process populates <confirmationId> element on the request message.

DVM-based translations:

- <serviceRequestType> - This element contains the Service Request Type code defined in the self-service application. It is translated using OTSS_ServiceRequestType DVM.
- <requestField> - This list contains field name/field value pairs and carry the details of the service request. It may include dates, codes, identifiers, such as collection notice ID, or tax type. The content of the <fieldValue> element may also belong to a lookup (predefined values list). The <fieldValue> node is translated using OTSS_Field Codes DVM, as follows:
 - A combination of [field name][separator][field value] from the self-service translated in to a [value] from the revenue management system. The separator is configured in *Service Configurations*.

The refund status (confirmation) details or the error message are translated following the *Common Mapping Rules*.

If integration encounters an exception (e.g., a connectivity or transformation error) while processing the message, integration returns a SOAP fault.

Integration Services

Name	Description
OTSSRefundStatusInquiryEBF	Refund Status Inquiry enterprise business flow process. The process will use the synchronous flow with confirmation ID. This process will have transformation using DVMs, pre- and post-extension services, and custom transformations.

Web Services

Name	Description
TSGetRefundStatus	This web service is used to process the input request details and check the status of the expected refund based on the information provided. The response contains refund status (confirmation) details and expected refund amounts.

Confirmation Inquiry by ID Integration Flow

Business Details

The taxpayer provides the confirmation ID as an input. The request is sent to the revenue management system and the response contains confirmation information.

Technical Details

This flow conforms to the *Synchronous Flow with Confirmation ID* pattern.

The self-service application sends the request in the form of an XML message to the integration. The integration forwards the request further to the revenue management system and gets the response that contains confirmation details.

The contents of the confirmation details or the error message are translated following the *Common Mapping Rules*.

If integration encounters an exception (e.g., a connectivity or transformation error) while processing the message, integration returns a SOAP fault.

Integration Services

Name	Description
OTSSRequestStatusInquiryEBF	Request Status Inquiry enterprise business flow process. The process uses the synchronous flow without confirmation ID. This process will have transformation using DVMs, pre- and post-extension services, and custom transformations.

Web Services

Name	Description
TSGetConfirmationInformation	This web service retrieves the confirmation details using the input confirmation ID.

Integration With Official Payments Corporation

This section describes the integration flows for the integration between Official Payments Corporation and the revenue management system.

Post Payment (Official Payments) Integration Flow

Business Details

The payment post-back XML is sent from Official Payments Corporation and received by a SOA composite process. The transaction is verified and then the message is transformed, assigned a confirmation ID, and transmitted to the revenue management system via a web service call.

The verification step provides additional protection against fraudulent payment submissions. The payment posting message contains a unique transaction ID that was initially generated by the integration and staged in the Payment Vendor Integration Reference table (*Prepare Payment Data Integration Flow*). This unique ID is transmitted to Official Payments Corporation when the payment process is redirected to their website, and is captured on the payment record in their system.

The payment post-back XML message is considered valid if the following conditions are satisfied:

- The unique ID on the message is not blank and the record for this unique ID exists in the staging table.
- The record in the staging table is in pending status, meaning this is the first attempt to post this specific payment.

Technical Details

This is an asynchronous flow without response.

Official Payments Corporation sends the Payment Posting information in the form of an XML message.

If the message does not contain a unique transaction ID (custom defined element, sequence 5 is empty) the integration sends an error notification e-mail and aborts further processing.

The message is transformed into a TSOneTimePayment request; the values are translated using DVMs as follows:

- `<payDestinationType>` - this element contains the Payment Destination code defined in the self-service application. It is translated using OTSS_PaymentDestination DVM.

- <paymentType> is translated using OPC_PaymentType DVM. The value represents source of the payment: checking or savings bank account, credit card or other. Official Payments Corp. provides the pre-defined list of payment type codes; these codes are translated into revenue management system's payment types.

The <currency> element is populated with the value of PaymentPostback.CurrencyCode property.

See "Official Payments Post-back XML Mapping" in [Appendix C](#) for more transformation details.

After transforming the XML, integration checks if the record with the request's external ID exists in the OTSS_PAYMENT_VENDOR_REF table.

- If the record exists and the status is pending (see ExternalPaymentData.Status.Pending property):
 - The integration process sets the status of the staging record to ExternalPaymentData.Status.Processed.
 - The integration process retrieves the taxpayer ID and "on behalf of" taxpayer ID from the staging record and populates the corresponding elements on the request message.
 - The integration process creates a confirmation ID as a concatenation of the prefix specified in ExternalPaymentData.ConfirmationID.Prefix property with the confirmation id coming from Official Payments and transformed into <extTransactionRefID>.

If the record does not exist, the integration sends an error notification e-mail and aborts further processing. The email notification configurations are defined in Module Configurations.

If payment posting has been verified successfully, the integration adds the message to Payment Posting JMS Queue.

The provider process picks the message from the Payment Posting queue and invokes revenue management system's TSOneTimePayment web service.

If the provider is not able to send the message to the revenue management system, the message is rolled back to the error queue the administrator can move the message back to the Main request queue from where it will be retried. See [How to Retry Technical Error Failure Messages](#) for more details.

Optional email notifications for the failure may be sent out from the integration layer (see [How To Configure Email Notification](#)).

Integration Services

Name	Description
OTSSPaymentPostingRequestEBF	Payment Posting enterprise business flow process. This process will have transformation using DVMs, pre- and post-extension services, and custom transformations. For Async flow, a JMS adapter will put the message in the Payment Posting queue.
OTSSPaymentPostingRequestProvider	Payment Posting provider process. SOA composite uses JMS adapter to read the message from Payment Posting queue and invokes ETPM XAI Inbound service. Consists of enrichment extension service and dynamic endpoint url. On error, the process will rollback the message to the error queue.

Adapter Services

Name	Description
OTSSPaymentPostingJMSProducer	OTSS Payment Posting JMS producer. JMS producer adapter to write to OTSS Payment Posting queue.
OTSSPaymentPostingJMSConsumer	OTSS Payment Posting JMS consumer. JMS consumer adapter to read from OTSS Payment Posting queue.
OTSSPaymentPostingDBAdapter	OTSS Payment Posting DB Adapter. DB adapter to query the OTSS_PAYMENT_VENDOR_REF to get the externalId.

JMS Queues

Name	Description
OTSSPaymentPosting	OTSS Payment Posting queue. Used by the integration layer to add transformed Payment Posting messages from OTSS.
OTSSPaymentPostingError	OTSS Payment Posting error queue. Error Queue for OTSS Payment Posting.

Web Services

Name	Description
TSOneTimePayment	See Payment Integration Flow

Process Payment Report (Official Payments) Integration Flow

Business Details

Official Payments Corporation emits the payment report (flat file, comma-delimited) to the customer (revenue authority) after finalizing the payment. The file is uploaded to the file server and picked up and transformed by the integration. The SOA composite process sends each report record as a web service request to the revenue management system for reconciliation.

Technical Details

Integration Services

Name	Description
OTSSReportReconciliationRequestEBF	Report Reconciliation enterprise business flow process. This process will have transformation using DVMS, pre- and post-extension services, and custom transformations. For Async flow, a JMS adapter will put the message in the Payment Posting queue.
OTSSReportReconciliationRequestProvider	Report Reconciliation provider process. SOA composite uses the JMS adapter to read the message from the Payment Report queue and invokes the ETPM XAI Inbound service. Consists of enrichment extension service and dynamic endpoint URL. On error, the process will rollback the message to the error queue.

Adapter Services

Name	Description
OTSSPaymentReportJMSProducer	OTSS Report Reconciliation JMS producer. JMS producer adapter to write to OTSS Report Reconciliation queue.

Name	Description
OTSSPaymentReportJMSConsumer	OTSS Report Reconciliation JMS consumer. JMS consumer adapter to read from OTSS Report Reconciliation queue.
OTSSReportReconciliationFileAdapter	OTSS Report Reconciliation File adapter. This file adapter will read the messages from the Payment Reconciliation Report sent by OPC.

JMS Queues

Name	Description
OTSSPaymentReport	OTSS Payment Report queue. Used by the integration layer to add transformed Payment Posting messages from OTSS.
OTSSPaymentReportError	OTSS Payment Report error queue. Error Queue for OTSS Payment Posting.

Web Services

Name	Description
TSPProcessExtPayReportRecord	Process External Payment Record enterprise business flow process. The process will use the synchronous flow with confirmation ID. This process will have transformation using DVMs, pre- and post-extension services, and custom transformations.

Monitoring the Integration

To monitor the integration flows use either one of the following methods:

- Monitoring the composite instances using WebLogic SOA Enterprise Manager.
- Monitoring the WebLogic logs.

Monitoring Using WebLogic SOA Enterprise Manager

1. Log in to the WebLogic SOA Server Enterprise Manager, and then navigate to SOA, SOA-Infra OTSS-ETPM. All composite processes deployed for integration are available under the partition OTSS-ETPM.
2. Select the appropriate process to list all the instances for the processes sorted by time of execution. The instances also have the request ID as part of the display name.
3. Click the appropriate process instance and it will display the flow for the process. The composite flow lists all activities in the process instance.

Monitoring Using WebLogic Logs

Log in to the machine where SOA server is installed. The SOA logs are stored in:

```
<WebLogic installation folder>/user_projects/domains/<SOA Domain name>/servers/<SOA Server name>/logs
```

Example:

```
/slot/ems1234/oracle/Middleware/user_projects/domains/soa_domain/servers/  
soa_server1/logs
```

Data Purge

To maintain maximum system integrity, the Oracle Fusion Middleware database should be purged periodically. For information about how to complete this task, refer to note 815896.1 on <https://support.oracle.com>.

Error Processing

The integration includes two types of errors:

This integration includes two types of errors:

Business Errors is a "valid" business rules-related error returned from the revenue management system (see "Error Message" in Appendix A: *Common XML Fragments*).

Technical Errors are triggered if integration encounters connectivity error, transformation error or other technical issue.

For asynchronous processes, the messages failed with technical errors are sent to the error queue and can be re-tried from integration layer. The error retry instructions and optional e-mail notification configuration are described below.

How to Configure Email Notification

- Log in to the Enterprise Manager console.
- Expand SOA and then right-click **SOA Infra**. From the menu, click **SOA Administration** and then click **Workflow Notification Properties**.
- From the drop-down list, select **EMAIL**.
- Enter the Email IDs in the From address field.

How to Retry Technical Error Failure Messages For Asynch Processes

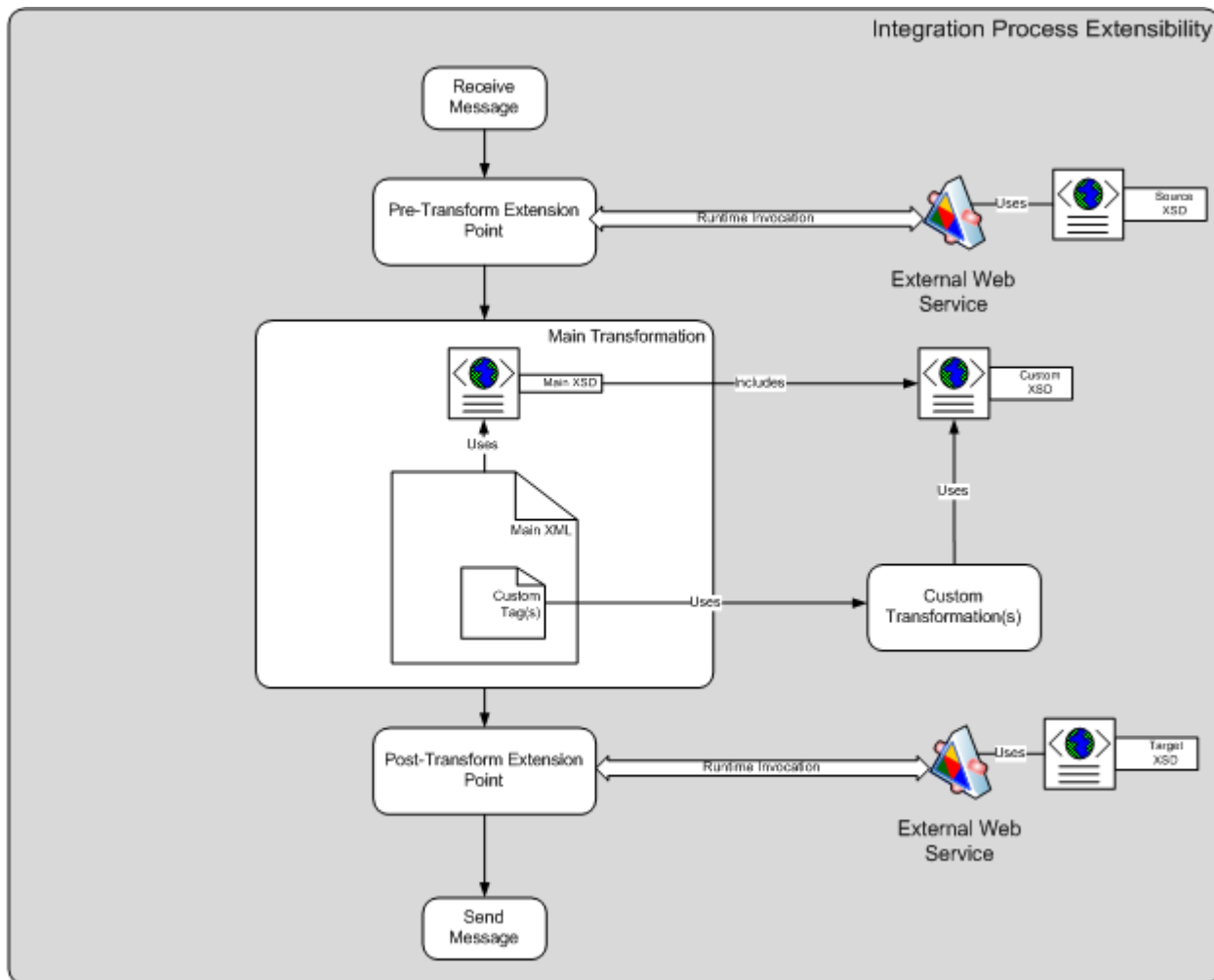
- In the WebLogic console, navigate to **Services > Messaging > JMS Modules**.
- Select the OTSS-ETPM Integration JMS Module to display all queues related to this integration.
- Select the appropriate error queue and click the Monitoring tab. This tab displays the details about messages in the queue in a table.
- Select the checkbox in the details table and click **Show Messages**. This displays all the messages in the error queue.
- Click **Move** and select **Move All**.
- Select the JMS server to move messages and then click **Next**.
- Select the correct parent queue for the error queue from the dropdown and click **Finish**.

This action moves all messages to the source queue, so that the integration layer processes all messages again.

Integration Flow	Type of Error	Action	Notification Type	Retry
Message sent by a Synchronous Request Flow from the Web Self Service to the Revenue Management system.	Business Error	Error is sent to the Web Self Service application	None	Retry can be triggered by the self service user

Integration Flow	Type of Error	Action	Notification Type	Retry
All processes except the asynchronous Taxpayer Service Request				
	Technical Error	Error is sent to the Web Self Service application		
Message sent using Asynchronous Request Flow from the Web Self Service application to the Revenue Management system. Asynchronous Taxpayer Service Request	Business Error	Message is moved into the error queue	Email notification (optional)	Retry can be triggered by the self service user
	Technical Error	Message is moved into the error queue	Email notification (optional)	Administrator has to move the messages from the error queue to the request queue from WebLogic Admin console.
Message sent using Asynchronous Request Flow from external application to the Revenue Management system Processes for the Integration with Official Payments	Business Error	Message is moved into the error queue	Email notification (optional)	Administrator has to move the messages from the error queue to the request queue from WebLogic Admin console
	Technical Error	Message is moved into the error queue	Email notification (optional)	Administrator has to move the messages from the error queue to the request queue from WebLogic Admin console.

Integration Extensibility



The typical Integration Process allows extensibility of the messages using three different methods:

- Pre-Transformation Extension Point
- Post-Transformation Extension Point
- Custom Transformations

In addition this integration offers an ability to customize the confirmation number generation logic and also features a dynamic end-point URL.

Important: Always create a back-up of your integration customizations before deploying a product update.

Pre-Transformation Extension Point

The pre-transformation extension point is invoked before the main transformation is executed. This transformation aids in transforming the source XML coming as an input to the integration process.

The integration layer defines an external call from the pre-transformation extension point. This extension point accepts source XML as input and gives the source XML as output. The implementation can choose to plug in a concrete WSDL instead of the abstract WSDL. This can assist the implementation in invoking any external web service and transform the input XML.

Post-Transformation Extension Point

The post-transformation extension point is invoked after the main transformation is executed. This transformation aids in transforming the target XML going as an input to the target queue.

The integration layer defines an external call from the post transformation extension point. This extension point accepts the target XML as input and gives the target XML as output. The implementation can choose to plug in a concrete WSDL instead of the abstract WSDL. This can assist the implementation in invoking any external web service and transform the output XML.

Custom Transformations

The custom transformations are used to add data to custom elements in the incoming and outgoing messages. The incoming and outgoing messages have custom elements defined in the message. These custom elements refer to a custom XML schema. The main transformation invokes custom transformation.

Empty custom transformation and custom schemas are shipped with the product. The implementation team can add additional fields in the custom schema and map them using the custom transformations.

Using custom transformations allows the implementation to define and pass additional data from the source system to the target system.

Dynamic End-Point URL

The end-point URL for all integration services is dynamic. They are derived at run-time from the **ConfigurationProperties.xml** file. This feature allows an implementation to support heterogeneous multi-system environments in which several back-end applications support separate functional areas. The end-point URL can, in fact, point to another custom web service instead of pointing to a back-end system web service. Use this option if a single request message needs to be routed to a similar back-end application. For example, if individual taxes are maintained in one system and the business taxes in another, the end-point URL for the refund status request could be a routing process that evaluates the input values and routes the request accordingly.

Open `apps/OTSS-ETPM/AIAMetaData/config/ConfigurationProperties.xml` and change the end-point URL of a specific service.

In the product install home, update MDS using the ant deploy command for Deploying MDS folder. For more information about the command to use to deploy to MDS, see "Deploying MDS Folder" in the Installation Guide.

After deploying the files to MDS, restart the SOA server.

Confirmation Number Utility

The confirmation number assignment can be customized for all or some of the integration flows.

- **Option 1.** Modify the default confirmation number prefix.

Locate the prefix property in the ConfigurationProperties.xml file (see [Service Configurations](#) for details) and change the value.

Modify the `$PRODUCT_HOME/ /services/industry/Tax/EBF/OTSSConfirmationIdService/xsl/Xform_GetConfirmationId.xsl`. Update the prefix check with the new value.

Redeploy the OTSSConfirmationIdService process using DeployComposite action. Refer to the Installation guide for more details.

- **Option 2.** Replace the confirmation number generation logic
Create your own confirmation number generation service(s) using the integration **OTSSConfirmationIdBPELProcess** wsdl located at:

```
$PRODUCT_HOME/MDS-Artifacts/OTSS-ETPM/AIAMetaData/AIAComponents/  
BusinessProcessServiceLibrary/OTSSConfirmationIdService/
```

Deploy this service at the location accessible from the SOA server.

In **ConfigurationProperties.xml** update the service-level properties for those services that should be using a new confirmation number utility service:

- [service name].ConfirmationNumberService.Name
 - [service name].ConfirmationNumberService.Port
 - [service name].ConfirmationNumberService.URL
- After updating files in the product install home, update MDS using the ant deploy command for Deploying MDS folder. For more information about the command to deploy to MDS, see "Deploying MDS Folder" in the Installation Guide.
After deploying the files to MDS, restart the SOA server.

Steps to Implement Extension Points

Each process in the integration has a pre- and post-transformation extension point which can be used to invoke Web services and transform the payload.

The desired extension point can be triggered from the process by enabling it using the **ConfigurationProperties.xml** pre- and post-transformation extension flags, as described in the [Setting Configuration Properties](#) section.

Each process has its own concrete wsdl which is used to read the endpoint location for the extension service.

These concrete wsdl files are located in MDS in the following directory:

```
/apps/ OTSS-ETPM/AIAMetaData/AIAComponents/ExtensionServiceLibrary/
```

Update the concrete wsdl file; modify **service soap:address location** to point to the URL of the extension service to be called, and move the concrete wsdl file to MDS.

To move the concrete wsdl to MDS, update the appropriate wsdl in the product install home. The directories to put the concrete wsdl in product install home are the following:

```
$PRODUCT_HOME/MDS-Artifacts/OTSS-ETPM
```

```
/AIAMetaData/AIAComponents/ExtensionServiceLibrary/
```

Then deploy the concrete wsdl to MDS by running the ant deploy command for Deploying MDS folder. For more information about the command to use to deploy to MDS, see "Deploying MDS Folder" the *Installation Guide*.

After deploying the files to MDS, restart the SOA server.

After restarting the SOA server, the extension point invokes the Web service in the concrete WSDL.

Example:

To enable the extension points for **OTSSPaymentExtension**, modify the service element in the **OTSSPaymentExtensionConcrete.wsdl**. In the following code, replace `<put your extension service URL here"/>` with the location of your extension service:

```
<service name="OTSSPaymentExtensionService">  
  <port name="OTSSPaymentV1ExtensionService_pt"  
    binding="otssext:OTSSPaymentV1ExtensionServiceBinding">
```

```
<!-- <soap:address location="http://burgb43102v.us.oracle.com:8001/soa-infra/
services/Extension/OTSSPaymentExtensionProcess/OTSSPaymentExtensionService"/>-->
<soap:address location="<put your extension service URL here"/>
</port>
</service>
```

Note: The wsdl service can be edited using Oracle JDeveloper 11g.

Steps to Implement Custom Transformations

Each process in the integration has its own XSD file.

The revenue management system – bound messages have custom elements which can be used to pass additional data.

Refer to message mappings to see the location of customElements in each message.

Each XSD has a corresponding CustomType xsd file in which the complexType elements for each customElements tag are defined.

To pass additional elements in the customElements tag, the corresponding complexType needs to be modified. Add the additional elements required in complexType element XSD.

Each process has a main transformation which invokes custom templates. Each main transformation file has a corresponding custom XSL and custom templates are defined in the custom XSL.

These custom templates are invoked at the location where each customElements tag is present.

The custom XSL can be modified to add transformation for the newly added elements in custom XSD files.

The custom XSD files are located in product install home under the following directories:

```
OTSSETPM/MDS-Artifacts/OTSS-ETPM/AIAMetaData/AIAComponents/
  ApplicationObjectLibrary/OTSS/V1/schemas
OTSSETPM/MDS-Artifacts/OTSS-ETPM/AIAMetaData/AIAComponents/
  ApplicationObjectLibrary/OPC/V1/schemas
...etc.
```

The custom XSL files are located in product install home under the directory:

```
OTSS-ETPM/services/industry/Utilities/EBF/<Process Name>/xsl
```

After updating the XSD and XSL files in the product install home, update MDS using the ant deploy command for Deploying MDS folder. For more information about the command to use to deploy to MDS, see "Deploying MDS Folder" in the Installation Guide.

After deploying the files to MDS, restart the SOA server.

After restarting the SOA server, the changes to the custom XSD and XSL will be reflected in the integration.

Chapter 9

Customizing the Portal Application

The self-service website is a Web Center application that comprises two types of pages:

- **Admin pages** that are used by implementers, administrators, and other users to configure the portal application. These pages are secured and are not intended to be exposed to taxpayers.
- **Public pages** that are used by taxpayers.

The most typical customization effort is likely to include:

- Configuring the Logo and Company tag line
- Configuring the portal custom icons and links
- Configuring the portal copyright message
- Adding a page to the portal
- Adding content to a portal page
- Changing the content of a portal page
- Changing label names
- Customizing Help content
- Changing the portal page template
- Changing the portal skin

This chapter starts with an exploration of two options for customizations:

- Using the WebCenter Application Override Bundle
- Using WebCenter Composer

This chapter also explores the incorporation of Universal Content Management (UCM) content in the portal pages.

For more information on WebCenter Composer and UCM, please refer to Oracle WebCenter documentation.

WebCenter Portal Application Override Bundle

The WebCenter Application Override Bundle is a file resource that is part of the WebCenter application framework. While this resource is used by WebCenter tools such as WebCenter Composer (described in the next section), it can also be used independently to support customizations without using additional tools.

The base product portal application, which is based on ADF and the WebCenter portal framework, references **Resource Bundles** for many portal-related elements, including:

- The text of UI elements on the portal pages (labels, titles, etc).
- The copyright message on the portal page template.
- Custom icons on the portal page template.

Resource Bundles are XML files (with an extension of .xlf) that hold values defined for Bundle IDs. For example:

```
<trans-unit id="BUNDLE_ID1">
  <source>Value for Bundle ID No. 1</source>
  <target/>
</trans-unit>
```

Resource bundles are used by the portal application at runtime to retrieve the values required for display or other purposes.

The base product contains many bundle files that are used to hold values for many components, with bundle IDs that are used throughout the application.

Resource bundle files are also language sensitive, which means that is the application supports multiple languages. There will be multiple resource bundles files covering the supported languages.

The base product also supports the use of an Application Override Bundle.

The Application Override Bundle is a special file that is used to provide override values for internal resource bundles. The override resource bundle is created automatically by the application framework and can be modified by implementers.

The base portal application uses the following logic to determine the value of a given resource bundle ID:

- If the bundle ID is found in the Application Override Bundle file, use the value from that file.
- Otherwise, use the value in the portal application internal resource bundle file(s).

The base product provides an Application Override Bundle Template File named WSSPortalOverrideBundle_ReferenceXLF.txt that includes the all the bundle IDs used by the portal application.

To change values of application elements that reference resource bundles, the following steps are required:

- Identify the resource bundle ID of the value to be changed – this depends on the type of element that has the bundle ID (label, text help, etc) and is described later in this chapter.
- Download the current Application Override Bundle file.
- Add the appropriate entries to the override bundle file (the bundle ID and value). For example:

```
<trans-unit id="BUNDLE_ID1">
  <source>Override value for Bundle ID No. 1</source>
  <target/>
</trans-unit>
```

- Upload the modified back bundle into the system.

Upload/Download of the Override Application Bundle

The application override bundle can be accessed through Oracle Enterprise Manager.

To download the override application bundle:

1. Log in into Oracle Enterprise Manager with the appropriate administrator user ID and password.
2. Use the export action (under MDS Configuration for the portal application) to export the MDS (Metadata Repository) of the portal application to a local file, typically an Archive file (.zip file).
3. Locate the Override Application Bundle file in the downloaded archive file. The name of the file will be **WSSPortalOverrideBundle_<language code>.xlf** (e.g., WSSPortalOverrideBundle_en.xlf for the English version).
4. Save the file locally for editing.

To upload the modified file back to the system:

1. Add the override bundle file back to the archive file (overriding the existing file in the archive).
2. Login into Oracle Enterprise Manager with the appropriate administrator user ID and password.
3. Use the import action (under MDS Configuration for the portal application) to import the archive file back to the MDS (Metadata Repository) of the portal.

For more information about MDS Configuration and Oracle Enterprise Manager, refer to the Oracle Enterprise Manager documentation.

WebCenter Composer and Administration Console

WebCenter Composer is a WebCenter tool that allows runtime customizations of a portal application. It allows users to modify existing portal pages or create new ones, all without changing the base product portal pages and structure.

The WebCenter Administration Console is a part of the WebCenter framework. It manages the application portal resources (such as pages, security information, etc.) and it invokes WebCenter Composer when editing or creating a new page.

To use the WebCenter Administration Console, implementers should be logged into the portal application using the admin login user ID and password.

The **Customization** link on the home page leads to the WebCenter Administration Console.

Administration Console Resources - Pages

In the admin console **Resources** tab implementers can find the **Pages** resource that lists all the portal pages. This page supports the following common actions:

The **Create Page** button will create a new page that will become a part of the portal application.

The **Show Page** checkbox on the pages list will determine whether the page is visible in the main navigation bar (and therefore if it is accessible independently without the need for hyperlinks).

The **Edit Page** action allows the implementer to modify the page definition and structure.

The **Set Access** action allows the implementer to define or change the access to that page based on the security rules and users defined in the system.

Important: Changing an existing page's access level can affect the base product portal behavior and is *not* recommended. *New* pages created from the admin console can be secured or made public as needed.

The **Delete Page** action allows the implementer to delete a page.

Important: Deleting base product pages will affect the base product portal behavior and is *not* recommended. New pages created from the admin console can be deleted as necessary.

Implementers can also change the order of the pages by using the **Move Page** action. This will change the order of pages in the portal main navigation bar.

For more information about the administration console and all the available actions please refer to Oracle WebCenter documentation.

Administration Console Resources - Page Templates

In the admin console **Resources** tab implementers can find the **Page Templates** resource that lists all the available page templates.

Implementers can **Create** new page templates, **Copy** an existing template or **Edit** a template in order to view or make changes.

Editing a page template can be done visually or by editing the actual page template code by using the **Edit Source** action.

Important: Changes to existing page templates can affect the way the base product portal behaves and therefore is *not* recommended. If additional template features are required for custom pages, implementers should create a new page template, copying the base template if appropriate, to avoid potential upgrade issues if base product templates change in future releases.

Administration Console Resources - Skins

In the admin console **Resources** tab implementers can find the **Skins** resource that lists all the available skins.

Implementers can **Create** new skins, **Copy** an existing skin or **Edit** a skin in order to view or make changes.

Editing a skin can be done visually or by editing the actual skin code by using the **Edit Source** action.

Important: Changes to existing skins can affect the way the base product portal behaves and should be done with care.

Administration Console Services - Content

In the admin console **Services** tab implementers can find the **Content** service that shows the configured content server linked to the portal application.

In this page implementers can view the folder structure and documents in the content server linked to the application.

Note: While some Content management functions are available in this page, the administration functions of the Content server are described in the UCM related sections.

Administration Console Configuration Page

The admin console **Configuration** allows implementers to make changes to default portal settings.

Implementers can change the **Default Page Template** which will change the page template on all portal pages.

The **Default Skin** for the portal pages can also be changed to provide a different look and feel to all pages at once.

Note: For more information about the actions and options available in the WebCenter Administration Console, please refer to Oracle WebCenter documentation.

Portal Page Link Reference

When adding content to the portal or referencing the portal from an external website there are a set of links that can be used to navigate to a base portal page.

The links will differ slightly depending on whether they are used in one of the portal pages or an external page or document.

Note: References from UCM documents are considered to be in the external pages category.

Portal Page Links From Within the Portal

The following are examples of links can be used internally within the portal.

Target	
Service Central page	/faces/oracle/webcenter/portalapp/pages/TaxpayerSvcReq.jspx
Service Central page for a specific service request	/faces/oracle/webcenter/portalapp/pages/TaxpayerSvcReq.jspx?serviceReq=<Service Request Type Code>
Interactive Tax Assistant (ITA) page	/faces/oracle/webcenter/portalapp/pages/IntgTaxAssist.jspx
ITA page for a specific interview	/faces/oracle/webcenter/portalapp/pages/IntgTaxAssist.jspx?interviewSetCd=<Interview Set Code>&interviewCd=<Interview Code>
Refund Status page	/faces/oracle/webcenter/portalapp/pages/RefundStatus.jspx
Refund Status page for a specific request	faces/oracle/webcenter/portalapp/pages/RefundStatus.jspx?refundReq=<Refund Request Code>
Payments page	/faces/oracle/webcenter/portalapp/pages/MakePayment.jspx

Portal Links from External Documents or Sites

To access a base portal page from an external document, web site or from a UCM document, the prefix "/etpmss" should be added to the link.

For example, the ITA page link will be:

```
/etpmss/faces/oracle/webcenter/portalapp/pages/IntgTaxAssist.jspx
```

Portal Links for New Portal Pages

When new pages are added into the portal via WebCenter Composer, these pages are given a URL by WebCenter. You can view the link to the page by selecting the **About This Page** action from the WebCenter administration console.

For example, a new page with an assigned URL of /oracle/webcenter/portalapp/pagehierarchy/Page1.jspx can be displayed from within the portal by using the assigned URL, or /faces/oracle/webcenter/portalapp/pagehierarchy/Page1.jspx, or by using the /etpmss/faces/oracle/webcenter/portalapp/pagehierarchy/Page1.jspx URL if accessed from a UCM document or external site.









Configuring the Logo and Company Tag Line

Changing the template page header and logo images is described in the following section about overriding portal application images.

The company tag line text that appears under the logo in the page template is derived from a resource bundle with the ID **TAG_LINE**. The text displayed for this ID can be modified using the same procedure used to modify any other resource bundle-based text values in the portal application (for more information, see the WebCenter Portal Application Override Bundle section of this document).

Overriding Portal Application Images

The base product portal application uses various images in its portal pages. These can be overridden by an implementer. The table below outlines the images that are used:

Image	File Name	Comments
	add.png	
	edit.png	
	delete.png	
	uapopup_sm_ena.png	This is the Online Help icon.
	ach-help.png	This image is used in the payment process.
	redirect-progress.gif	This image is used in the payment process, when the user is redirected to the OPC website.
	wsslogo.png	This is the organization Logo image displayed at the top of each page.
	wssheader.png	This is the background image that is display at the top of each page.

All images used in the portal application come from the WSSExtensionWar.war file.

To override a base product image:

1. Using WebLogic Console, stop or undeploy the portal application.
2. Get a copy of the WSSExtensionWar.war (this file is supplied with the application installation package).
3. Copy a new image file with the appropriate name (according to the table above) into the .war file (overriding the existing image file).
4. Deploy the .war file back to the server as a Shared Library. After deployment, this file can be located on the WebLogic server as **oracle.otss.extension(1.0,1.0)**.

For more information about deployment of the portal application refer to the WebLogic documentation and the Installation Guide.

Configuring the Portal Custom Icons and Links

The portal custom icons area is a horizontal bar on the page template that can contain images with links to common websites, such as Facebook, YouTube, Twitter, etc.

The portal custom links area is set of hyperlinks on the page template that can contain links to common pages both inside and outside the portal. Examples include an agency contact information page, agency general information, privacy policy link, and more.

Defining Portal Custom Icons

The base product portal application supports up to five custom icons defined by implementers.

To define a custom icon, implementers should define the image that will be displayed and the URL to use when the user clicks on the image.

All definitions are done using the resource bundle values in the Application Override Bundle with the following bundle ids:

- **COMMONICON n .IMAGE** – for the image reference.
- **COMMONICON n .LINK** – for the URL definition.

n can be 1 to 5.

The following example is for an icon definition for YouTube:

```
<trans-unit id="COMMONICON1.IMAGE">
  <source>/images/youtube.png</source>
  <target/>
</trans-unit>
```

```
<trans-unit id="COMMONICON1.LINK">
  <source>http://www.youtube.com</source>
  <target/>
</trans-unit>
```

As a general guideline, the typical icon image size is 32x32 pixels.

Important: The icon image should be copied into the WSSExtensionWar.war file so it can be referenced as "/images/...".

Defining Portal Custom Links

The base product portal application supports up to 10 links defined by implementers. The base product is configured with a default set of sample links. These default links can be changed or removed by implementers.

In order to define a custom link, implementers should define the link name and the destination URL for that link.

All the definitions are done using resource bundle values in the Application Override Bundle using the following bundle IDs:

- **COMMONLINK n .IMAGE**
- **COMMONLINK n .DESTINATION**

n can be 1 to 10.

The following is an example of a link definition:

```
<trans-unit id="COMMONLINK1">
  <source>Contact Us</source>
  <target/>
</trans-unit>
<trans-unit id="COMMONLINK1.DESTINATION">
  <source>http://www.YourCompanyContactInfo.com</source>
  <target/>
</trans-unit>
```

Configuring the Portal Copyright Message

The base product portal includes a copyright message, defined at the bottom of the portal page template. This message has a default value but can be changed by implementers.

The copyright message definition is resource bundle values in the Application Override Bundle using the following bundle ID: **COPYRIGHT_MSG**.

Adding a New Page to the Portal

Adding a new page to the portal is done using the Create Page action in the WebCenter administration console. The new page that is created can be based on any generic page template that is available, the base product page template, or any new page template that was created by the implementer.

The new page can include any content that can be added using WebCenter Composer and can have links to external pages or existing portal pages.

The new page can be secured or unsecured. If it is defined as visible, it will automatically be shown on the portal main navigation bar if the user has the appropriate security access.

Adding Content to a Portal Page

Many of the base product portal pages have some reserved space for specific portal content. This portal content is likely to change from one implementation to another.

Many types of content can be added, including HTML documents, images, links, custom developed control, and more.

Adding content to a portal page is done by using WebCenter Composer. In order to add something to a portal page the page has to be edited by using the **Edit Page** action in the WebCenter administration console. When the page is in edit mode, using WebCenter Composer, implementers can add content in the areas that are available for modification on the page.

Areas available for adding content will typically show as a dashed-line box with the caption, "Add Content".

When the area for adding content is chosen, the type of content to add can be selected from the available components in WebCenter Composer.

Adding WebCenter Managed Content to a Portal Page

WebCenter provides the ability to manage documents in a document management repository called WebCenter Content Server (UCM). These documents can be any documents, text or binary, HTML or images.

WebCenter Content Server provides the ability to securely manage the documents, providing a version control mechanism such as check-in and check-out capabilities.

Documents stored and managed by WebCenter Content Server can be included on the base product or new portal pages.

For more information about WebCenter Managed Content and UCM, refer to the WebCenter documentation.

In order to add a UCM document to a portal page, the **Content Management** category should be selected (after selecting the **Add Content** action) and within that category the Content Presenter can be selected.

Note that there are many ways to include UCM documents and other WebCenter Content features in a page. For more information about that, refer to WebCenter documentation.

When adding a UCM document using the **Content Presenter Component**, the user should select a document stored in one of the WebCenter Content server **Folders**.

For more information about Setting up UCM folders, refer to the WebCenter documentation.

Including Images and References in UCM Documents

UCM documents that will be included in the portal may include attachments that will be uploaded to UCM as well and should be accessible from the portal page that includes the UCM main document.

The following guidelines ensure that attachments (e.g., images) of a UCM documents can be visible on the portal page if presented using the **Content Presenter** component:

- Both document and attachments should be uploaded into UCM into the same or different UCM folders. All UCM folders would reside under the main **Contribution Folders** folder.
- If the document includes references to the attachments (for example if the document is an HTML fragment that includes images then the images are the attachments that are referenced inside that document) the following path should be used:

```
/etpmss/content/conn/UCM/path/Contribution%20Folders/<directory name>
```

- The **<directory name>** is the UCM folder that holds attachment files. In case of an HTML document the above link should be used as a Relative Location in the document that it is included in, in order to avoid the need to provide the name of the server or port number.
- Using the above link will ensure that the attachment that is stored in UCM is correctly displayed in the portal page.

Changing Content of a Portal Page

Implementers can choose to make certain changes in the base provided portal pages. Some of the typical changes include adding content to the page, changing the look-and-feel of certain component on the page and changing text on the page (e.g., labels or headings).

All these changes are done using WebCenter Composer after selecting the **Edit Page** action from the WebCenter administration console.

Changes made using WebCenter Composer can be revoked so that the page returns to its original setup. The revocation of changes can be done at a component or page level.

For more information on how to use WebCenter Composer to make and revoke changes in page elements, refer to the WebCenter documentation.

While implementers can use WebCenter Composer to make changes to many page elements, there are some changes that can be done using a different technique. These are changes that are either more global in nature such as changing many labels or help text items, all of which are explored in the following sections.

Changing Page Labels

Most of the base product labels on the portal pages can be changed. Labels can be field labels, titles, headings or any static text that is displayed on a page. Label text values are derived from resource bundles based on a bundle ID. Updating the page labels can be done in two ways:

- Change the label text via WebCenter Composer.
- Change the label text in the Application Override Bundle.

Identifying the Bundle ID for a Label on a Page

The bundle ID for a label can be located either in the application override template file (provided with the base product), according to the naming convention listed below, or by using WebCenter Composer, following these steps:

1. Login to the application as an administrator and navigate to the WebCenter Administration Console – Pages.
2. Select **Edit Page** from the **Action** dropdown for the page where the label content is placed.
3. Select **View > Source** and navigate to the label.

Note: The easiest way to find the component in source mode is to click on the component in the design/preview section. For pages that are visible only after navigation, first navigate to the page and then change view to Source. You may be asked to confirm the Edit inside task flow; if so, select **Edit**.

4. Click on the label or the component that contains the label and click on edit in toolbar, a popup will be launched. Go the display options tab and copy the value for the **Text/Label** property. This will represent an EL expression in which second part is the bundle id for example **TS_ADMIN_MANAGELANG_LANGUAGE_ADMIN_LBL**.

Note: Sometimes the label itself cannot be selected, but the component that contains it will allow changing it.

Label Bundle ID Format

Base product bundle IDs have the following naming convention:

- For Admin functionality (e.g., Lookup admin, Service request admin, etc):

```
TS_Admin_<Page or Fragment file name>_<Component Name>_LBL
```

.

- For the rest:

```
TS_<Page or Fragment file name>_<Component Name>_LBL
```

.

All label bundle IDs have the prefix `TS_` and the suffix `_LBL`.

If the label belongs to an admin module, the prefix is followed by the keyword `ADMIN`.

If the label is part of a transaction page, the page name or page fragment name directly follows the prefix.

For a complete list of all bundle IDs, refer to the application override bundle template file that is provided with the base product.

Changing Label Text

Label text can either be changed using the application override bundle or through WebCenter Composer.

Changing the Label Text Value Using Application Override Bundle

Modifying the help text in the Application Override Bundle is done according to the Label resource bundle ID. Once the identifier of the bundle is located, an entry can be added to the override bundle file.

Refer to the [WebCenter Portal Application Override Bundle](#) section for more details about the procedure of overriding resource bundle values in the base portal application.

Changing the Label Text Value Using WebCenter Composer

Modifying the label text using WebCenter Composer involves the following steps:

1. Follow the steps to locate the bundle ID of the label (using WebCenter Composer) on the page that needs to be changed (refer to the [Identifying the Bundle ID for a Label on a Page](#) topic).
2. After locating the label and editing it to see the bundle ID, you can update the Text/Label property with the new value.

The Text/Label property contains the text to be displayed and can include a simple text or a resource bundle reference. It is recommended to use WebCenter Composer to create a New bundle ID entry in the portal application override bundle with the new text that needs to be displayed. This new bundle ID will be specific to the implementation.

Referencing a resource bundle value will allow implementers to retain the multi-language support for this label.

Customizing Help Content

The base product portal application provides context specific help in all the admin pages.

All transactional pages have placeholders for help which are hidden by default. The typical customizations of help component include:

- Change the help text for any or all help components.
- Hide an existing help component.
- Show existing help components that were formerly hidden.

Help Component Structure

All help components are isolated ADF task flows with the following input parameters:

bypassBundleLookup	Specifies if the help text is to be taken from the bundle or from the string parameter passed in helpText. Possible values are true or false . true – skip the bundle and display text string provided. false – find text in the bundle.
helpText	Required when bypassBundleLookup is true . String provided will be displayed as help for the given component.
bundleId	Required when bypassBundleLookup is false . Help text against this bundle ID will be displayed in UI.
visible	Possible value is true or false . Specifies if the help text needs to be displayed or hidden.

Base product help text is derived from the resource bundle and is referenced by a bundle ID. This bundle ID can be used to override the help text value in the application override bundle.

Identifying the Bundle ID for a Help Component on a Page

The bundle ID for a Help component can be located either in the application override template file (provided with the base product), according to the naming convention listed below, or by using WebCenter Composer, following these steps:

1. Login to the application as an administrator and navigate to the WebCenter Administration Console – Pages.
2. Select **Edit Page** from the **Action** dropdown for the page where the help content is placed.
3. Select **View > Source** and navigate to the help component (task flow).

The easiest way to find the component in source mode is to click on the component in the design/preview section. For pages that are visible only after navigation, first navigate to the page and then change view to Source. You may be asked to confirm the Edit inside task flow; if so, select **Edit**.

4. In source view select the task flow region and click on edit in toolbar, a popup will be launched. Go the parameters tab and copy the value for Bundle ID parameter. In the EL expression (the value that was copied), the *second* part of the expression is the bundle ID. For example: `TS_ADMIN_MANAGEMSG_MAIN_HELP`.

Help Bundle ID Format

Base product bundle IDs have the following naming convention:

- For Admin functionality (e.g., Lookup admin, Service request admin, etc):

```
TS_Admin_<Page or Fragment file name>_<Component Name>_HELP
```

- For the rest:

```
TS_<Page or Fragment file name>_<Component Name>_HELP
```

All help bundle IDs have the prefix `TS_` and the suffix `_HELP`.

If the help belongs to an admin module, the prefix is followed by the keyword `ADMIN`.

If the help is part of a transaction page, the page name or page fragment name directly follows the prefix.

For a complete list of all bundle IDs, refer to the application override bundle template file that is provided with the base product.

Changing Help Text

Help text can either be changed using the application override bundle or through WebCenter Composer.

Changing the Help Text Using Application Override Bundle

Modifying the help text in the Application Override Bundle is done according to the help text resource bundle ID. Once the identifier of the bundle is located, an entry can be added to the override bundle file.

Refer to the [WebCenter Portal Application Override Bundle](#) section for more details about the procedure of overriding resource bundle values in the base portal application.

Changing the Help Text Using WebCenter Composer

Modifying the help text using WebCenter Composer involves the following steps:

- Follow the steps to located the bundle id of the help component (using WebCenter Composer) of the page that needs to be changed (refer to [Identifying the Bundle ID for a Help Component on a Page](#) section).
- After locating the help component and editing it to see the bundle ID, you can update the help component properties (described in the [Help Component Structure](#) section).

The `helpText` property contains the text to be displayed and can include simple text or a resource bundle reference. It is recommended to use WebCenter Composer to create a New bundle ID entry in the portal application override bundle with the new text that needs to be displayed. This new bundle ID will be specific to the implementation.

Referencing a resource bundle value will allow implementers to retain the multi-language support for the Help text.

Hide/Show a Help Component on a Page

Base product is provided with help components that are visible and some that are hidden.

Follow the steps below to hide and/or show Help on pages.

1. Follow the WebCenter Composer steps to located the bundle ID of the help component of the page that needs to be changed (refer to *Identifying the Bundle ID for a Help Component on a Page* section).
2. After locating the help component and editing it to see the bundle ID, use Web Center Composer to update the help component visible property (described in the Help Component Structure section).

Changing Portal Page Template

A base product template page is supplied with the portal application. This template page can be copied or modified by using WebCenter Composer. Implementers can update the page template visually or by directly accessing and updating the template source code.

Important: If more advanced customizations, beyond what is described in this chapter, are required, it is recommended that the base product page template is copied and changes are made on the new customized template.

If a new page template is created, implementers can switch all or some of the existing pages, include the base product pages, to use the new template. This is done by using the WebCenter administration console configuration page.

Changing Portal Skin

The base product includes a default skin **WSSPortalTemplate_v1** that includes the following aliases:

WSSRightOuterLayout	Alias associated with right hand layout of the page
WSSPageTitle	Alias associated with title of every page.
WSSHelpImage	Alias associated with help image.
WSSSubHeaderLayout	Alias associated with the subheader layout.
WSSSubHeader	Alias associated with subheader.
WSSCommonIcon	Alias associated the images in footer bar area.
WSSSiteBackgroundColor	Alias associated with background color of the self-service application.

Updating the skin for update can be done by using WebCenter Composer. Implementers can update the skin visually or by directly accessing and updating the skin source code.

Important: If skin changes are required, it is recommended that the base product skin is copied and changes are made on the new customized skin.

Managing Portal Customization

Portal customizations that are done for one operational environment may be needed in other environments as well. For example, customization done in development environment can be moved to the testing environment before approval and finally moved to the production environment.

In addition, portal customization should be preserved as much as possible when a new product release is installed.

Porting Portal Customizations from One Environment to Another

In order to move all the portal customizations from environment A to environment B, the following steps are required:

- Install the application on environment B, as per the base product installation documentation.
- Using Oracle Enterprise Manager in environment A, export the MDS (Metadata Repository) of the portal application to a local file (leave the “Exclude base documents” option UNCHECKED).
- Using Oracle Enterprise Manager in environment B, Import the MDS from the local file to the MDS of the portal application.

For more information about MDS Configuration and Oracle Enterprise Manager, refer to the Oracle Enterprise Manager documentation.

Retaining Portal Customization After New Product Release Installation

When installing a new product version, there are no specific steps that have to be taken to retain the portal customization.

If the MDS repository of the existing application remains intact and the new portal version is installed and deployed on top of the existing portal, all portal customization should persist.

Important: Importing (using Oracle Enterprise Manager) MDS data from a previous version product environment into a new install product version is NOT recommended since it can override the new version portal pages.

For more information about MDS, refer to WebCenter and WebLogic documentation.

Moving UCM Content from One Environment to Another

Documents managed via WebCenter Content Server can be referenced on portal pages.

Given that a typical installation of the base product includes a separate UCM repository for each environment, the UCM document should be moved between environments when the application is moved.

Moving UCM documents between UCM environments is done using the WebCenter Content Server Archiver process.

Moving documents between UCM environments involves the following steps:

- Setting up an Archive process in UCM in environment A:
 - Defining an Archive name and UCM folders associated with it.
 - Defining the export criteria.
- Exporting the content to a file on the UCM server in environment A (by using the Export action of the Archiver).
- Transferring the file to correct UCM server location in environment B (e.g., using FTP).
- Importing the content from the file to the UCM repository in environment B (by using the Import action of the Archiver).

For more information about the WebCenter Archiver and the process of exporting UCM data, refer to the system migration and archiving in the WebCenter and WebCenter Content Server documentation.

Appendix A

WSDL Library

Generic Service Request - TSTaxpayerServiceRequest

Node	Description
<TSTaxpayerServiceRequest>	This is the main node of the Generic Service Request WSDL. dateTimeTagFormat property value is set to 'xsd'.
<head>	This node is a sub node of <TSTaxpayerServiceRequest> It contains the access keys and audit information of the request. Refer to Access Keys and Audit Node.
<requestStatus> <errorMessage>	These nodes are sub nodes of <TSTaxpayerServiceRequest> Refer to Error Message Node for details
<confirmationData>	This node is a sub node of <TSTaxpayerServiceRequest>. Refer to Confirmation Message Node for details
<mainData>	This node is a sub node of <TSTaxpayerServiceRequest>. It contains the main data of the request.
<serviceRequestType>	This node is a sub node of <mainData>. Service Request Type code, alphanumeric.
<responseMode>	This node is a sub node of <mainData>. It holds the mode of response that is associated with the request. Valid values: SYNCH (Synchronous), ASYNCH (Asynchronous)
<serviceRequestData> <requestField> <sequence> <fieldName>	This node is a sub node of <mainData>. It holds a list of service request fields. requestField – a list of service request fields sequence contains the field's sequence number. fieldName contains the name of a field.

Node	Description
<fieldValue>	fieldValue contains the value of a field.
</requestField>	
</serviceRequestData>	
<customInfo>	This node is a sub node of <TSTaxpayerServiceRequest>. Contains generic name/value collection. This is a placeholder for additional information that can be utilized by the implementation. Refer to Custom Information Node for details.

Taxpayer Identification Request - TSTaxpayerIdentification

Node	Description
<TSTaxpayerIdentification>	This is the main node of the Taxpayer Identification Service Request WSDL. dateTimeTagFormat property value is set to 'xsd'..
<head>	This node is a sub node of <TSTaxpayerIdentification>. It contains the access keys and the audit information of the request. Refer to Access Keys and Audit Node.
<requestStatus>	These nodes are sub nodes of <TSTaxpayerIdentification>. Refer to Error Message Node for details
<errorMessage>	
<confirmationData>	This node is a sub node of <TSTaxpayerIdentification>. Refer to Confirmation Message Node for details
<mainData>	This node is a sub node of <TSTaxpayerIdentification>. It contains the main data of the request.
<serviceRequestType>	This node is a sub node of <mainData>. Service Request Type code, alphanumeric.
<responseMode>	This node is a sub node of <mainData>. It holds the mode of response that is associated with the request. Valid values: SYNCH (Synchronous), ASYNCH (Asynchronous)
<serviceRequestData>	This node is a sub node of <mainData>. It holds a list of service request fields.
<requestField>	requestField – a list of service request fields
<sequence>	sequence contains the field's sequence number.
<fieldName>	fieldName contains the name of a field.
<fieldValue>	fieldValue contains the value of a field.
</requestField>	
</serviceRequestData>	
<customInfo>	This node is a sub node of <TSTaxpayerIdentification>. Contains generic name/value collection. This is a placeholder for additional information that can be utilized by the implementation. Refer to Custom Information Node for details.
<responseData>	This node is a sub node of <TSTaxpayerIdentification>. It holds response details from the web service.
<responseDetails>	responseDetails – the list of taxpayer ID-s identified based on the input request data information.
<sequence>	sequence contains the response details' sequence number.
<taxpayerId>	

Node	Description
<code></responseDetails></code>	taxpayerId contains the Taxpayer ID retrieved based on the query.
<code></responseData></code>	

Refund Inquiry Request - TSGetRefundStatus

Node	Description
<code><TSGetRefundStatus></code>	This is the main node of the Refund Inquiry Service Request WSDL. dateTimeTagFormat property value is set to 'xsd'
<code><head></code>	This node is a sub node of <code><TSGetRefundStatus></code> . It contains the access keys and the audit information of the request. Refer to Access Keys and Audit Node.
<code><requestStatus></code> <code><errorMessage></code>	These nodes are sub nodes of <code><TSGetRefundStatus></code> . Refer to Error Message Node for details
<code><confirmationData></code>	This node is a sub node of <code><TSTaxpayerIdentification></code> . Refer to Confirmation Message Node for details
<code><mainData></code>	This node is a sub node of <code><TSGetRefundStatus></code> . It contains the main data of the request.
<code><serviceRequestType></code>	This node is a sub node of <code><mainData></code> . Service Request Type code, alphanumeric.
<code><serviceRequestData></code> <code><requestField></code> <code><sequence></code> <code><fieldName></code> <code><fieldValue></code> <code></requestField></code> <code></serviceRequestData></code>	This node is a sub node of <code><mainData></code> . It holds a list of service request fields. responseDetails – the list of taxpayer ID-s identified based on the input request data information. sequence contains the response details' sequence number. taxpayerId contains the Taxpayer ID retrieved based on the query.
<code><responseData></code> <code><responseDetails></code> <code><expectedDate></code> <code><refundAmount></code> <code><carryOverAmount></code> <code><offsetAmount></code> <code></responseDetails></code> <code></responseData></code>	This node is a sub node of <code><TSGetRefundStatus></code> . It holds response details from the web service. responseDetails contains the details of the refund that will be returned to the taxpayer. expectedDate contains the Expected Refund Date. refundAmount contains the amount of refund that will be returned. carryOverAmount contains the carry over amount (if there's any). offsetAmount contains the offset amount (if there's any).
<code><customInfo></code>	This node is a sub node of <code><TSTaxpayerIdentification></code> . Contains generic name/value collection. This is a placeholder for additional information that can be utilized by the implementation. Refer to Custom Information Node for details.

One Time Payment - TSOneTimePayment

Node	Description
<TSONeTimePayment>	This is the main node of the One Time Payment WSDL. dateTimeTagFormat property value is set to 'xsd'
<head>	This node is a sub node of <TSONeTimePayment>. It contains the access keys and the audit information of the request. Refer to Access Keys and Audit Node.
<requestStatus> <errorMessage>	These nodes are sub nodes of <TSONeTimePayment>. Refer to Error Message Node for details
<confirmationData>	This node is a sub node of <TSTaxpayerIdentification>. Refer to Confirmation Message Node for details
<mainData>	This node is a sub node of <TSONeTimePayment>. It contains the main data of the request.
<payDestinationType>	This node is a sub node of <mainData>. Contains Payment Destination code for the payment. Defines the purpose of the payment, configured in self-service application
<destinationDetails> <sequence> <fieldName> <fieldValue> </destinationDetails>	This node is a sub node of <mainData>. Each sub node holds the information of a specific field destination in the request. sequence contains the field's sequence number. fieldName contains the name of a field. fieldValue contains the value of a field.
<paymentType>	This node is a sub node of <mainData> contains the payment type code, identifies the payment instruments
<amount>	This node is a sub node of <mainData> contains payment amount
<currency>	This node is a sub node of <mainData> the currency of the payment.
<paymentDate>	This node is a sub node of <mainData>. Contains payment date.
<bankAcctInfo> <routingNumber> <acctNumber> </bankAcctInfo>	This node is a sub node of <mainData>. It holds the information regarding the bank account. routingNumber contains the routing number of the bank acctNumber contains taxpayer account number
<extTransactionRefID>	This node is a sub node of <mainData>. It holds the reference id of the payment made via external payment services provider. This node may contain credit card authorization code, receipt number or any other transaction identifier.
<contactEmailAddress>	This node is a sub node of <mainData>. It holds the value of the email address of the taxpayer.
<externalId>	This node is a sub node of <mainData>. It holds the unique id generated to trace the interaction with external payment services system.
<paymentVendor>	This node is a sub node of <mainData>. It holds the identifier of the payment vendor that processed the payment.
<externalPaymentData>	This node is a sub node of <mainData>. It holds vendor-specific information for the payment made using external payment services.
<customInfo>	This node is a sub node of <TSTaxpayerIdentification>. Contains generic name/value collection. This is a placeholder for additional information that can be utilized by the implementation. Refer to Custom Information Node for details.

Prepare Payment Data - TSPPrepareExtPaymentData

Node	Description
<TSPPrepareExtPaymentData>	This is the main node of the Prepare External Payment Data WSDL. dateTimeTagFormat property value is set to 'xsd'
<head>	This node is a sub node of <TSPPrepareExtPaymentData>. It contains the access keys and the audit information of the request. The valid values for the <action> element are: VALIDATEONLY and PREPARE. Refer to Access Keys and Audit Node for more details.
<requestStatus> <errorMessage>	These nodes are sub nodes of <TSPPrepareExtPaymentData>. Refer to Error Message Node for details
<confirmationData>	This node is a sub node of <TSPPrepareExtPaymentData>. Common fragment, not in use for this web service
<mainData>	This node is a sub node of <TSPPrepareExtPaymentData>. It contains the main data of the request.
<taxpayerID>	This node is a sub node of <mainData>. Contains internal taxpayer identifier in the revenue management system.
<onBehalfOfTaxpayerID>	This node is a sub node of <mainData>. Contains internal taxpayer identifier in the revenue management system.
<paymentVendor>	This node is a sub node of <mainData>. It holds the identifier of the payment vendor that processed the payment.
<amount>	This node is a sub node of <mainData>. Contains payment amount.
<paymentDate>	This node is a sub node of <mainData>. Contains payment date.
<payDestinationType>	This node is a sub node of <mainData>. Contains Payment Destination code for the payment. Defines the purpose of the payment, configured in self-service application
<destinationDetails> <sequence> <fieldName> <fieldValue> </destinationDetails>	This node is a sub node of <mainData>. Each sub node holds the information of a specific field destination in the request. sequence contains the field's sequence number. fieldName contains the name of a field. fieldValue contains the value of a field.
<paymentDetails> <sequence> <fieldName> <fieldValue> </paymentDetails>	This node is a sub node of <mainData>. Contains payment-related information prepared in the revenue management system for subsequent transmission to external payment services provider. sequence contains the detail's sequence number. fieldName contains the name of a field. fieldValue contains the value of a field.
<externalId>	This node is a sub node of <mainData>. It holds the unique id generated to trace the interaction with external payment services system.
<feeRequirement>	This node is a sub node of <mainData>. It contains the indicator that determines whether and how convenience fees will be collected for this payment. Applicable for payments made via external payment services providers.

Node	Description
<customInfo>	This node is a sub node of <TSPrepareExtPaymentData>. Contains generic field name/field value pairs collection. This is a placeholder for additional information that can be utilized by the implementation. Refer to Custom Information Node for details.

Process Reconciliation Report - TSPProcessExtPayReportRecord

Node	Description
<TSPProcessExtPayReportRecord>	This is the main node of the One Time Payment WSDL. dateTimeTagFormat property value is set to 'xsd'
<head>	This node is a sub node of <TSPProcessExtPayReportRecord>. It contains the access keys and the audit information of the request. Refer to Access Keys and Audit Node for more details..
<requestStatus> <errorMessage>	These nodes are sub nodes of <TSPProcessExtPayReportRecord>. Refer to Error Message Node for details
<confirmationData>	This node is a sub node of <TSPProcessExtPayReportRecord>. Common fragment, not in use by this web service
<paymentReportRecord>	This node is a sub node of <TSPProcessExtPayReportRecord>. It contains the main data of the request.
<paymentVendor>	This node is a sub node of <mainData>. It holds the identifier of the payment vendor that processed the payment.
<amount>	This node is a sub node of <mainData>. Contains payment amount.
<paymentDate>	This node is a sub node of <mainData>. Contains payment date.
<payDestinationType>	This node is a sub node of <mainData>. Contains Payment Destination code for the payment. Defines the purpose of the payment, configured in self-service application
<destinationDetails> <sequence> <fieldName> <fieldValue> </destinationDetails>	This node is a sub node of <mainData>. Each sub node holds the information of a specific field destination in the request. sequence contains the field's sequence number. fieldName contains the name of a field. fieldValue contains the value of a field.
<paymentType>	This node is a sub node of <mainData> contains the payment type code, identifies the payment instruments
<extTransactionRefID>	This node is a sub node of <mainData>. It holds the reference id of the payment made via external payment services provider. This node may contain credit card authorization code, receipt number or any other transaction identifier.
<externalId>	This node is a sub node of <mainData>. It holds the unique id generated to trace the interaction with external payment services system.
<externalPaymentReportData>	This node is a sub node of <mainData> It holds vendor-specific payment information.

Node	Description
<customInfo>	This node is a sub node of <TSProcessExtPayReportRecord>. It holds the details of any customization incorporated into the web service. Refer to Custom Information Node.

Confirmation Inquiry - TSGetConfirmationInformation

Node	Description
<TSGetConfirmationInformation>	This is the main node of the Confirmation Inquiry WSDL. dateTimeTagFormat property value is set to 'xsd'.
<head>	This node is a sub node of <TSGetConfirmationInformation>. It contains the access keys and the audit information of the request. Refer to Access Keys and Audit Node.
<requestStatus> <errorMessage>	These nodes are sub nodes of <TSGetConfirmationInformation>. Refer to Error Message Node for more details.
<confirmationData>	This node is a sub node of <TSGetConfirmationInformation>. Contains confirmation ID, confirmation header and te list of confirmation details. For detailed description refer to Confirmation Message Node.

Common XML Fragments

These fragments are included in every XML message transmitted from the self-service application.

Access Keys and Audit

This is a common node that contains the access keys and audit information of the request.

Node	Description
<head>	Request audit info: accessing user details an access keys
<action>	This node is a sub node of <head>. It indicates the specific business purpose of the request. Optional.
<key1> <name> <value/> </key1>	This node is a sub node of <head>. It contains the access key for the service. Most often the access to the revenue management data is keyed by a taxpayer ID (key name=PER_ID)
<key2> <name> <value> </key2>	This node is a sub node of <head>. Contains the access key for the service: account ID, taxpayer ID, tax type etc.
<key3> <name> <value> </key3>	This node is a sub node of <head>. Contains the access key for the service: account ID, taxpayer ID, tax type etc.

Node	Description
<key4> <name> <value> </key4>	This node is a sub node of <head>. Contains the access key for the service: account ID, taxpayer ID, tax type etc.
<key5> <name> <value> </key5>	This node is a sub node of <head>. Contains the access key for the service: account ID, taxpayer ID, tax type etc.
<webUserId>	This node is a sub node of <head>. It contains the User ID of the online user. For casual (not logged in) web user, the value defaulted to anonymous .
<webUserName>	This node is a sub node of <head>. It contains the User Name of the online user. For casual (not logged in) web user, the value defaulted to anonymous .
<emailAddress>	This node is a sub node of <head>. It contains the Email Address of the online user.
<ipAddress>	This node is a sub node of <head>. It contains the Internet Protocol (IP) Address of the computer terminal used by the online user.

Error Message

This is a common fragment that contains error message details.

Node	Description
<requestStatus>	This node indicates the status of the request. Evaluated by the SOA composite processes in the integration layer.
<errorMessage>	This node contains error message information.
<messageCategory>	messageCategory contains the Message Category number of the error.
<messageNumber>	messageNumber contains the error Message Number
<messageParameters>	messageParameters Message Parameters of the error
<parameters>	parameters parameters list
<sequence>	sequence contains the sequence number of the parameter.
<parameterType>	parameterType contains the type of the parameter.
<parameterValue>	Valid values: DATE (Date), STRING (String), NUMBER (Number) or CURRENCY (Currency).
<parameters>	parameterValue contains the value of the parameter.
<messageParameters>	currency contains the currency code to be used in the message when translating parameters of type CURRENCY
<currency>	messageTxtOvrd contains the override message text.
<messageTxtOvrd>	
<errorMessage>	

Confirmation Message

This is a common node that holds the confirmation info returned upon successful processing of the request.

Node	Description
<confirmationData>	This node contains the confirmation information
<confirmationId>	This node is a sub node of <confirmationData>. It contains Confirmation ID of the request returned by the web service.
<header>	This node is a sub node of <confirmationData>. It contains a list of the header information of the confirmation message.
<messageCategory>	messageCategory contains the Message Category number of the confirmation.
<messageNumber>	messageNumber contains the Message Number
<messageParameters>	messageParameters contains Message Parameters
<parameters>	parameters parameters list
<sequence>	sequence contains the sequence number of the parameter.
<parameterType>	parameterType contains the type of the parameter.
<parameterValue>	Valid values: DATE (Date), STRING (String), NUMBER (Number) or CURRENCY (Currency).
</parameters>	
</messageParameters>	
<currency>	parameterValue contains the value of the parameter.
<messageTxtOvrd>	currency contains the currency code to be used in the message when translating parameters of type CURRENCY
</header>	messageTxtOvrd contains the override message text.
<details>	This node is a sub node of <confirmationData>. It contains a list of confirmation details (messages).
<messageCategory>	messageCategory contains the Message Category number of the confirmation.
<messageNumber>	messageNumber contains the Message Number
<messageParameters>	messageParameters contains Message Parameters
<parameters>	parameters parameters list
<sequence>	sequence contains the sequence number of the parameter.
<parameterType>	parameterType contains the type of the parameter.
<parameterValue>	Valid values: DATE (Date), STRING (String), NUMBER (Number) or CURRENCY (Currency).
</parameters>	
<currency>	parameterValue contains the value of the parameter.
<messageTxtOvrd>	currency contains the currency code to be used in the message when translating parameters of type CURRENCY
</details>	messageTxtOvrd contains the override message text.

Custom Information

This is a common node that contains a generic field name/field value pair's collection. It can be used by various customization extension components for information exchange.

Node	Description
<customInfo>	This node contains a list of generic field name/value pairs. It is a placeholder for the data populated by various customization extensions.
<sequence>	sequence contains the sequence number of the entry.
<fieldName>	fieldName contains the name of the field.
<fieldValue>	fieldValue contains the value
</customInfo>	

Appendix B

Sample Messages

GetConfirmationID

GetConfirmationID Request

```
<part name="payload"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <process xmlns="http://xmlns.oracle.com/OTSS/OTSSConfirmationIdService/
OTSSConfirmationIdBPELProcess">
    <input name="PaymentSequence.Prefix">SRID</input>
  </process>
</part>
```

GetConfirmationID Response

```
<processResponse xmlns:client="http://xmlns.oracle.com/OTSS/
OTSSConfirmationIdService/OTSSConfirmationIdBPELProcess"
xmlns="http://xmlns.oracle.com/OTSS/OTSSConfirmationIdService/
OTSSConfirmationIdBPELProcess">
  <client:result>SRID81000000</client:result>
</processResponse>
```

GetRefundStatus

GetRefundStatus Request

```
<ns1:TSGetRefundStatus xmlns:ns1="http://oracle.com/TSGetRefundStatus.xsd">

  <ns1:head>
    <ns1:action />
    <ns1:webUserId>anonymous</ns1:webUserId>
    <ns1:webUserName>anonymous</ns1:webUserName>
    <ns1:emailAddress />
    <ns1:ipAddress>10.159.122.216</ns1:ipAddress>
  </ns1:head>
  <ns1:mainData>
    <ns1:serviceRequestType>
      REFUND_STATUS_UNR</ns1:serviceRequestType>
    <ns1:serviceRequestData>
      <ns1:requestField>
        <ns1:sequence>1</ns1:sequence>
        <ns1:fieldName>ENTITY_NAME</ns1:fieldName>
        <ns1:fieldValue>Hansen Peter</ns1:fieldValue>
      </ns1:requestField>
      <ns1:requestField>
        <ns1:sequence>2</ns1:sequence>
        <ns1:fieldName>ID_TYPE</ns1:fieldName>
        <ns1:fieldValue>DVL</ns1:fieldValue>
      </ns1:requestField>
      <ns1:requestField>
        <ns1:sequence>3</ns1:sequence>
        <ns1:fieldName>ID_VALUE</ns1:fieldName>
        <ns1:fieldValue>123-456-009</ns1:fieldValue>
      </ns1:requestField>
      <ns1:requestField>
        <ns1:sequence>4</ns1:sequence>
        <ns1:fieldName>TAX_TYPE</ns1:fieldName>
        <ns1:fieldValue>IND-INCOME</ns1:fieldValue>
      </ns1:requestField>
      <ns1:requestField>
        <ns1:sequence>5</ns1:sequence>
        <ns1:fieldName>FILING_START_DATE</ns1:fieldName>
        <ns1:fieldValue>2010-01-01</ns1:fieldValue>
      </ns1:requestField>
      <ns1:requestField>
        <ns1:sequence>6</ns1:sequence>
        <ns1:fieldName>FILING_END_DATE</ns1:fieldName>
        <ns1:fieldValue>2010-12-31</ns1:fieldValue>
      </ns1:requestField>
      <ns1:requestField>
        <ns1:sequence>7</ns1:sequence>
        <ns1:fieldName>EXPECTED_REFUND</ns1:fieldName>
        <ns1:fieldValue>4628</ns1:fieldValue>
      </ns1:requestField>
    </ns1:serviceRequestData>
  </ns1:mainData>
</ns1:TSGetRefundStatus>
```

GetRefundStatus Response

```
<TSGetRefundStatus xmlns:ns0="http://oracle.com/TSGetRefundStatus.xsd"
dateTimeTagFormat=""
xmlns="http://oracle.com/TSGetRefundStatus.xsd">
  <ns0:head>
    <ns0:action />
    <ns0:key1>
      <ns0:name />
      <ns0:value />
    </ns0:key1>
    <ns0:key2>
      <ns0:name />
      <ns0:value />
    </ns0:key2>
    <ns0:key3>
      <ns0:name />
      <ns0:value />
    </ns0:key3>
    <ns0:key4>
      <ns0:name />
      <ns0:value />
    </ns0:key4>
    <ns0:key5>
      <ns0:name />
      <ns0:value />
    </ns0:key5>
    <ns0:webUserId>anonymous</ns0:webUserId>
    <ns0:webUserName>anonymous</ns0:webUserName>
    <ns0:emailAddress />
    <ns0:ipAddress>10.159.122.216</ns0:ipAddress>
  </ns0:head>
  <ns0:requestStatus />
  <ns0:confirmationData>
    <ns0:confirmationId>RSID66000000</ns0:confirmationId>
    <ns0:header>
      <ns0:messageCategory>11126</ns0:messageCategory>
      <ns0:messageNumber>13001</ns0:messageNumber>
      <ns0:messageParameters />
      <ns0:messageTxtOvrd />
    </ns0:header>
  </ns0:confirmationData>
  <ns0:mainData>
    <ns0:serviceRequestType>
      SS-REFUND-STATUS</ns0:serviceRequestType>
    <ns0:serviceRequestData>
      <ns0:requestField>
        <ns0:sequence>1</ns0:sequence>
        <ns0:fieldName>ENTITY_NAME</ns0:fieldName>
        <ns0:fieldValue>Hansen Peter</ns0:fieldValue>
      </ns0:requestField>
      <ns0:requestField>
        <ns0:sequence>2</ns0:sequence>
        <ns0:fieldName>ID_TYPE</ns0:fieldName>
        <ns0:fieldValue>DVL</ns0:fieldValue>
      </ns0:requestField>
      <ns0:requestField>
        <ns0:sequence>3</ns0:sequence>
        <ns0:fieldName>ID_VALUE</ns0:fieldName>
```

```

    <ns0:fieldValue>123-456-009</ns0:fieldValue>
  </ns0:requestField>
  <ns0:requestField>
    <ns0:sequence>4</ns0:sequence>
    <ns0:fieldName>TAX_TYPE</ns0:fieldName>
    <ns0:fieldValue>IND-INCOME</ns0:fieldValue>
  </ns0:requestField>
  <ns0:requestField>
    <ns0:sequence>5</ns0:sequence>
    <ns0:fieldName>FILING_START_DATE</ns0:fieldName>
    <ns0:fieldValue>2010-01-01</ns0:fieldValue>
  </ns0:requestField>
  <ns0:requestField>
    <ns0:sequence>6</ns0:sequence>
    <ns0:fieldName>FILING_END_DATE</ns0:fieldName>
    <ns0:fieldValue>2010-12-31</ns0:fieldValue>
  </ns0:requestField>
  <ns0:requestField>
    <ns0:sequence>7</ns0:sequence>
    <ns0:fieldName>EXPECTED_REFUND</ns0:fieldName>
    <ns0:fieldValue>4628</ns0:fieldValue>
  </ns0:requestField>
</ns0:serviceRequestData>
</ns0:mainData>
<ns0:responseData>
  <ns0:refundAmount>309.00</ns0:refundAmount>
  <ns0:expectedDate>2011-02-03</ns0:expectedDate>
</ns0:responseData>
</TSGetRefundStatus>

```

IdentifyTaxpayer

IdentifyTaxpayer Request

```

<ns1:TSTaxpayerIdentification xmlns:ns1="http://oracle.com/
TSTaxpayerIdentification.xsd">
  <ns1:head>
    <ns1:webUserId>anonymous</ns1:webUserId>
    <ns1:webUserName>anonymous</ns1:webUserName>
    <ns1:ipAddress>10.159.240.238</ns1:ipAddress>
  </ns1:head>
  <ns1:mainData>
    <ns1:serviceRequestType>
      IDENTIFY_TAX_CLERANCE</ns1:serviceRequestType>
    <ns1:responseMode>SYNCH</ns1:responseMode>
    <ns1:serviceRequestData>
      <ns1:requestField>
        <ns1:sequence>1</ns1:sequence>
        <ns1:fieldName>ENTITY_NAME</ns1:fieldName>
        <ns1:fieldValue>Hansen Peter</ns1:fieldValue>
      </ns1:requestField>
      <ns1:requestField>
        <ns1:sequence>2</ns1:sequence>
        <ns1:fieldName>BIRTH_DATE</ns1:fieldName>
        <ns1:fieldValue>1965-01-20</ns1:fieldValue>
      </ns1:requestField>
    </ns1:serviceRequestData>
  </ns1:mainData>
</ns1:TSTaxpayerIdentification>

```

```

    <ns1:sequence>3</ns1:sequence>
    <ns1:fieldName>TAXPAYER_TYPE</ns1:fieldName>
    <ns1:fieldValue>IND</ns1:fieldValue>
  </ns1:requestField>-
  <ns1:requestField>
    <ns1:sequence>4</ns1:sequence>
    <ns1:fieldName>ID_TYPE</ns1:fieldName>
    <ns1:fieldValue>DVL</ns1:fieldValue>
  </ns1:requestField>-
  <ns1:requestField>
    <ns1:sequence>5</ns1:sequence>
    <ns1:fieldName>ID_VALUE</ns1:fieldName>
    <ns1:fieldValue>123-567-009</ns1:fieldValue>
  </ns1:requestField></ns1:serviceRequestData>
</ns1:mainData>
</ns1:TSTaxpayerIdentification>

```

IdentifyTaxpayer Response

```

<TSTaxpayerIdentification xmlns:p="http://schemas.oracle.com/service/bpel/common"
xmlns:ns4="http://ouaf.oracle.com/spl/XAIXapp/xaiserver/
OTSSIdentifyTaxpayerExtension/V1"
xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
xmlns:ns0="http://oracle.com/TSTaxpayerIdentification.xsd"
dateTimeTagFormat=""
xmlns="http://oracle.com/TSTaxpayerIdentification.xsd">
  <ns0:head>
    <ns0:action />
    <ns0:key1>
      <ns0:name />
      <ns0:value />
    </ns0:key1>
    <ns0:key2>
      <ns0:name />
      <ns0:value />
    </ns0:key2>
    <ns0:key3>
      <ns0:name />
      <ns0:value />
    </ns0:key3>
    <ns0:key4>
      <ns0:name />
      <ns0:value />
    </ns0:key4>
    <ns0:key5>
      <ns0:name />
      <ns0:value />
    </ns0:key5>
    <ns0:webUserId>anonymous</ns0:webUserId>
    <ns0:webUserName>anonymous</ns0:webUserName>
    <ns0:emailAddress />
    <ns0:ipAddress>10.159.240.238</ns0:ipAddress>
  </ns0:head>
  <ns0:requestStatus />
  <ns0:mainData>
    <ns0:serviceRequestType>
      IDENTITY-CHECK-TAX-CLR</ns0:serviceRequestType>
    <ns0:responseMode>SYNCH</ns0:responseMode>
    <ns0:serviceRequestData>

```

```

<ns0:requestField>
  <ns0:sequence>1</ns0:sequence>
  <ns0:fieldName>ENTITY_NAME</ns0:fieldName>
  <ns0:fieldValue>Hansen Peter</ns0:fieldValue>
</ns0:requestField>
<ns0:requestField>
  <ns0:sequence>2</ns0:sequence>
  <ns0:fieldName>BIRTH_DATE</ns0:fieldName>
  <ns0:fieldValue>1965-01-20</ns0:fieldValue>
</ns0:requestField>
<ns0:requestField>
  <ns0:sequence>3</ns0:sequence>
  <ns0:fieldName>TAXPAYER_TYPE</ns0:fieldName>
  <ns0:fieldValue>INDIVIDUAL</ns0:fieldValue>
</ns0:requestField>
<ns0:requestField>
  <ns0:sequence>4</ns0:sequence>
  <ns0:fieldName>ID_TYPE</ns0:fieldName>
  <ns0:fieldValue>DVL</ns0:fieldValue>
</ns0:requestField>
<ns0:requestField>
  <ns0:sequence>5</ns0:sequence>
  <ns0:fieldName>ID_VALUE</ns0:fieldName>
  <ns0:fieldValue>123-456-009</ns0:fieldValue>
</ns0:requestField>
</ns0:serviceRequestData>
</ns0:mainData>
<ns0:responseData>
  <ns0:responseDetails>
    <ns0:sequence>1</ns0:sequence>
    <ns0:taxpayerId>9047106386</ns0:taxpayerId>
  </ns0:responseDetails>
</ns0:responseData>
</TSTaxpayerIdentification>

```

OneTimePayment

OneTimePayment Request

```

<ns1:TSTimePayment xmlns:ns1="http://oracle.com/TSTimePayment.xsd">
  <ns1:head>
    <ns1:action />
    <ns1:key1>
      <ns1:name>PER_ID</ns1:name>
      <ns1:value>6888315916</ns1:value>
    </ns1:key1>
    <ns1:webUserId>anonymous</ns1:webUserId>
    <ns1:webUserName>anonymous</ns1:webUserName>
    <ns1:emailAddress>john.hanses@gmail.com</ns1:emailAddress>
    <ns1:ipAddress>10.159.101.209</ns1:ipAddress>
  </ns1:head>
  <ns1:mainData>
    <ns1:payDestinationType>COLL_NOTICE</ns1:payDestinationType>
    <ns1:destinationDetails>
      <ns1:sequence>1</ns1:sequence>
    </ns1:destinationDetails>
  </ns1:mainData>
</ns1:TSTimePayment>

```

```

    <ns1:fieldName>COLLECTION_NOTICE_NO</ns1:fieldName>
    <ns1:fieldValue>6888315374</ns1:fieldValue>
  </ns1:destinationDetails>
  <ns1:paymentType>CHECKING</ns1:paymentType>
  <ns1:amount>100</ns1:amount>
  <ns1:currency>USD</ns1:currency>
  <ns1:paymentDate>2012-10-01-04:00</ns1:paymentDate>
  <ns1:bankAcctInfo>
    <ns1:routingNumber>321171184</ns1:routingNumber>
    <ns1:acctNumber>111111</ns1:acctNumber>
  </ns1:bankAcctInfo>
  <ns1:contactEmailAddress>
    john.hanses@gmail.com</ns1:contactEmailAddress>
  <ns1:paymentVendor>ETPMSS</ns1:paymentVendor>
</ns1:mainData>
</ns1:TSTimePayment>

```

OneTimePayment Response

```

<TSTimePayment xmlns:ns0="http://oracle.com/TSTimePayment.xsd"
dateTimeTagFormat=" "
xmlns="http://oracle.com/TSTimePayment.xsd">
  <ns0:head>
    <ns0:action />
    <ns0:key1>
      <ns0:name>PER_ID</ns0:name>
      <ns0:value>6888315916</ns0:value>
    </ns0:key1>
    <ns0:key2>
      <ns0:name />
      <ns0:value />
    </ns0:key2>
    <ns0:key3>
      <ns0:name />
      <ns0:value />
    </ns0:key3>
    <ns0:key4>
      <ns0:name />
      <ns0:value />
    </ns0:key4>
    <ns0:key5>
      <ns0:name />
      <ns0:value />
    </ns0:key5>
    <ns0:webUserId>anonymous</ns0:webUserId>
    <ns0:webUserName>anonymous</ns0:webUserName>
    <ns0:emailAddress>john.hansen@gmail.com</ns0:emailAddress>
    <ns0:ipAddress>10.159.101.209</ns0:ipAddress>
  </ns0:head>
  <ns0:requestStatus />
  <ns0:errorMessage />
  <ns0:confirmationData>
    <ns0:confirmationId>PTID61000000</ns0:confirmationId>
    <ns0:header>
      <ns0:messageCategory>11126</ns0:messageCategory>
      <ns0:messageNumber>11004</ns0:messageNumber>
      <ns0:messageParameters />
      <ns0:messageTxtOvrd />
    </ns0:header>
  </ns0:confirmationData>
</TSTimePayment>

```



```

</ns0:header>
<ns0:details>
  <ns0:sequence>1</ns0:sequence>
  <ns0:messageCategory>11126</ns0:messageCategory>
  <ns0:messageNumber>11005</ns0:messageNumber>
  <ns0:messageParameters>
    <ns0:parameters>
      <ns0:sequence>1</ns0:sequence>
      <ns0:parameterValue>$100.00</ns0:parameterValue>
    </ns0:parameters>
    <ns0:parameters>
      <ns0:sequence>2</ns0:sequence>
      <ns0:parameterValue>10-01-2012</ns0:parameterValue>
    </ns0:parameters>
  </ns0:messageParameters>
  <ns0:messageTxtOvrdr />
</ns0:details>
</ns0:confirmationData>
<ns0:mainData>
  <ns0:payDestinationType>COLL_NOTICE</ns0:payDestinationType>
  <ns0:destinationDetails>
    <ns0:sequence>1</ns0:sequence>
    <ns0:fieldName>COLLECTION_NOTICE_NO</ns0:fieldName>
    <ns0:fieldValue>6888315374</ns0:fieldValue>
  </ns0:destinationDetails>
  <ns0:paymentType>CHECKING</ns0:paymentType>
  <ns0:amount>100</ns0:amount>
  <ns0:currency>USD</ns0:currency>
  <ns0:paymentDate>2012-10-01</ns0:paymentDate>
  <ns0:bankAcctInfo>
    <ns0:routingNumber>321171184</ns0:routingNumber>
    <ns0:acctNumber>111111</ns0:acctNumber>
  </ns0:bankAcctInfo>
  <ns0:extTransactionRefID />
  <ns0:contactEmailAddress>
    john.hansen@gmail.com</ns0:contactEmailAddress>
  </ns0:mainData>
</TSTimePayment>

```

PrepareExtPaymentData

PrepareExtPaymentData Request

```

<ns1:TSPrepareExtPaymentData xmlns:ns1="http://oracle.com/
TSPrepareExtPaymentData.xsd">
  <ns1:head>
    <ns1:action>VALIDATEONLY</ns1:action>
    <ns1:key1>
      <ns1:name>PER_ID</ns1:name>
      <ns1:value>6888315916</ns1:value>
    </ns1:key1>
    <ns1:webUserId>anonymous</ns1:webUserId>
    <ns1:webUserName>anonymous</ns1:webUserName>
    <ns1:emailAddress />
    <ns1:ipAddress>10.154.132.208</ns1:ipAddress>
  </ns1:head>

```

```

<ns1:mainData>
  <ns1:amount>200</ns1:amount>
  <ns1:paymentDate>2012-10-02</ns1:paymentDate>
  <ns1:payDestinationType>COLL_NOTICE</ns1:payDestinationType>
  <ns1:destinationDetails>
    <ns1:sequence>1</ns1:sequence>
    <ns1:fieldName>COLLECTION_NOTICE_NO</ns1:fieldName>
    <ns1:fieldValue>6888315374</ns1:fieldValue>
  </ns1:destinationDetails>
</ns1:mainData>
</ns1:TSPPrepareExtPaymentData>

```

PrepareExtPaymentData Response

```

<TSPPrepareExtPaymentData xmlns:p="http://schemas.oracle.com/service/bpel/common"
xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
xmlns:ns1="http://schemas.oracle.com/service/bpel/common"
xmlns:client="http://oracle.com/TSPPrepareExtPaymentData.xsd"
dateTimeFormat=""
xmlns="http://oracle.com/TSPPrepareExtPaymentData.xsd">
  <client:head>
    <client:action>VALIDATEONLY</client:action>
    <client:key1>
      <client:name>PER_ID</client:name>
      <client:value>6888315916</client:value>
    </client:key1>
    <client:key2 />
    <client:key3 />
    <client:key4 />
    <client:key5 />
    <client:webUserId>JohnSmith</client:webUserId>
    <client:webUserName>JohnSmith</client:webUserName>
    <client:emailAddress />
    <client:ipAddress>10.154.132.208</client:ipAddress>
  </client:head>
  <client:errorMessage />
  <client:confirmationData>
    <client:header>
      <client:messageCategory />
      <client:messageParameters />
      <client:messageTxtOvrld />
    </client:header>
  </client:confirmationData>
  <client:mainData>
    <client:taxpayerID>6888315916</client:taxpayerID>
    <client:amount>200</client:amount>
    <client:paymentDate>2012-10-02</client:paymentDate>
    <client:payDestinationType>
    COLL_NOTICE</client:payDestinationType>
    <client:destinationDetails>
      <client:sequence>1</client:sequence>
      <client:fieldName>COLLECTION_NOTICE_NO</client:fieldName>
      <client:fieldValue>6888315374</client:fieldValue>
    </client:destinationDetails>
    <client:externalID>
    0915acde-5f95-4dfd-8497-511137b02ald</client:externalID>
    <client:feeRequirement />
  </client:mainData>

```

```
</TSPrepareExtPaymentData>
```

RequestStatusInquiry

RequestStatusInquiry Request

```
<ns1:TSGetConfirmationInformation dateTimeTagFormat="xsd"
xmlns:ns1="http://oracle.com/TSGetConfirmationInformation.xsd">
  <ns1:head>
    <ns1:key1 />
    <ns1:key2 />
    <ns1:key3 />
    <ns1:key4 />
    <ns1:key5 />
    <ns1:webUserId>anonymous</ns1:webUserId>
    <ns1:webUserName>anonymous</ns1:webUserName>
    <ns1:ipAddress>10.154.186.249</ns1:ipAddress>
  </ns1:head>
  <ns1:errorMessage>
    <ns1:messageParameters>
      <ns1:parameters />
    </ns1:messageParameters>
  </ns1:errorMessage>
  <ns1:confirmationData>
    <ns1:confirmationId>SRID10000000</ns1:confirmationId>
    <ns1:header>
      <ns1:messageParameters>
        <ns1:parameters />
      </ns1:messageParameters>
    </ns1:header>
    <ns1:details>
      <ns1:messageParameters>
        <ns1:parameters />
      </ns1:messageParameters>
    </ns1:details>
  </ns1:confirmationData>
</ns1:TSGetConfirmationInformation>
```

RequestStatusInquiry Response

```
<TSGetConfirmationInformation xmlns:ns0="http://oracle.com/
TSGetConfirmationInformation.xsd"
dateTimeTagFormat="xsd"
xmlns="http://oracle.com/TSGetConfirmationInformation.xsd">
  <ns0:head>
    <ns0:action />
    <ns0:key1>
      <ns0:name />
      <ns0:value />
    </ns0:key1>
    <ns0:key2>
      <ns0:name />
      <ns0:value />
    </ns0:key2>
```

```

<ns0:key3>
  <ns0:name />
  <ns0:value />
</ns0:key3>
<ns0:key4>
  <ns0:name />
  <ns0:value />
</ns0:key4>
<ns0:key5>
  <ns0:name />
  <ns0:value />
</ns0:key5>
<ns0:webUserId>anonymous</ns0:webUserId>
<ns0:webUserName>anonymous</ns0:webUserName>
<ns0:emailAddress />
<ns0:ipAddress>10.154.186.249</ns0:ipAddress>
</ns0:head>
<ns0:requestStatus />
<ns0:errorMessage>
  <ns0:currency />
</ns0:errorMessage>
<ns0:confirmationData>
  <ns0:confirmationId>SRID10000000</ns0:confirmationId>
  <ns0:header>
    <ns0:messageCategory>11126</ns0:messageCategory>
    <ns0:messageNumber>10001</ns0:messageNumber>
    <ns0:messageParameters />
    <ns0:messageTxtOvrdr />
  </ns0:header>
  <ns0:details>
    <ns0:sequence>1</ns0:sequence>
    <ns0:messageCategory>11126</ns0:messageCategory>
    <ns0:messageNumber>15012</ns0:messageNumber>
    <ns0:messageParameters>
      <ns0:parameters>
        <ns0:sequence>1</ns0:sequence>
        <ns0:parameterValue>
          peter.hansen@gmail.com</ns0:parameterValue>
        </ns0:parameters>
      </ns0:messageParameters>
      <ns0:messageTxtOvrdr />
    </ns0:details>
  </ns0:confirmationData>
</TSGetConfirmationInformation>

```

TaxClearanceCertificate

TaxClearanceCertificate Request

```

<ns1:TSTaxpayerServiceRequest xmlns:ns1="http://oracle.com/
TSTaxpayerServiceRequest.xsd">
  <ns1:head>
    <ns1:key1>
      <ns1:name>PER_ID</ns1:name>
      <ns1:value>9047106386</ns1:value>
    </ns1:key1>

```

```

<ns1:webUserId>anonymous</ns1:webUserId>
<ns1:webUserName>anonymous</ns1:webUserName>
<ns1:ipAddress>10.159.122.216</ns1:ipAddress>
</ns1:head>
<ns1:mainData>
  <ns1:serviceRequestType>
    TAX_CLEARANCE_CERT</ns1:serviceRequestType>
  <ns1:responseMode>SYNCH</ns1:responseMode>
  <ns1:serviceRequestData>
    <ns1:requestField>
      <ns1:sequence>1</ns1:sequence>
      <ns1:fieldName>EMAIL_ADDRESS</ns1:fieldName>
      <ns1:fieldValue>peter.hansen@gmail.com</ns1:fieldValue>
    </ns1:requestField>
    <ns1:requestField>
      <ns1:sequence>2</ns1:sequence>
      <ns1:fieldName>PHONE_NUMBER</ns1:fieldName>
      <ns1:fieldValue>415 235-3421</ns1:fieldValue>
    </ns1:requestField>
    <ns1:requestField>
      <ns1:sequence>3</ns1:sequence>
      <ns1:fieldName>TAX_CERT_PURPOSE</ns1:fieldName>
      <ns1:fieldValue>PERSONAL</ns1:fieldValue>
    </ns1:requestField>
    <ns1:requestField>
      <ns1:sequence>4</ns1:sequence>
      <ns1:fieldName>PAPER_COPY_REQUIRED</ns1:fieldName>
      <ns1:fieldValue>>true</ns1:fieldValue>
    </ns1:requestField>
    <ns1:requestField>
      <ns1:fieldName>Please provide additional comments
      here</ns1:fieldName>
      <ns1:fieldValue>Requried for Government
      Grant</ns1:fieldValue>
    </ns1:requestField>
  </ns1:serviceRequestData>
</ns1:mainData>
</ns1:TSTaxpayerServiceRequest>

```

TaxClearanceCertificate Response

```

<TSTaxpayerServiceRequest xmlns:params="http://schemas.oracle.com/service/bpel/
common"
xmlns:otss="http://oracle.com/TSTaxpayerServiceRequest.xsd"
dateTimeTagFormat=" "
xmlns="http://oracle.com/TSTaxpayerServiceRequest.xsd">
  <otss:head>
    <otss:key1>
      <otss:name>PER_ID</otss:name>
      <otss:value>9047106386</otss:value>
    </otss:key1>
    <otss:key2 />
    <otss:key3 />
    <otss:key4 />
    <otss:key5 />
    <otss:webUserId>anonymous</otss:webUserId>
    <otss:webUserName>anonymous</otss:webUserName>
    <otss:ipAddress>10.159.122.216</otss:ipAddress>

```

```

</otss:head>
<otss:errorMessage>
  <otss:currency />
  <otss:messageTxtOvrdr />
</otss:errorMessage>
<otss:confirmationData>
  <otss:confirmationId>SRID81000000</otss:confirmationId>
  <otss:header>
    <otss:messageCategory>11126</otss:messageCategory>
    <otss:messageNumber>10001</otss:messageNumber>
    <otss:messageTxtOvrdr />
  </otss:header>
  <otss:details>
    <otss:sequence>1</otss:sequence>
    <otss:messageCategory>11126</otss:messageCategory>
    <otss:messageNumber>15010</otss:messageNumber>
    <otss:messageParameters>
      <otss:parameters>
        <otss:sequence>1</otss:sequence>
        <otss:parameterType>DATE</otss:parameterType>
        <otss:parameterValue>2012-10-02</otss:parameterValue>
      </otss:parameters>
    </otss:messageParameters>
    <otss:messageTxtOvrdr />
  </otss:details>
</otss:confirmationData>
<otss:mainData>
  <otss:serviceRequestType>
    TAX-CLR-CERTIFICATE</otss:serviceRequestType>
  <otss:responseMode>SYNCH</otss:responseMode>
  <otss:serviceRequestData>
    <otss:requestField>
      <otss:sequence>1</otss:sequence>
      <otss:fieldName>EMAIL_ADDRESS</otss:fieldName>
      <otss:fieldValue>peter.hansen@oracle.com</otss:fieldValue>
    </otss:requestField>
    <otss:requestField>
      <otss:sequence>2</otss:sequence>
      <otss:fieldName>PHONE_NUMBER</otss:fieldName>
      <otss:fieldValue>415 235-3421</otss:fieldValue>
    </otss:requestField>
    <otss:requestField>
      <otss:sequence>3</otss:sequence>
      <otss:fieldName>TAX_CERT_PURPOSE</otss:fieldName>
      <otss:fieldValue>C1P</otss:fieldValue>
    </otss:requestField>
    <otss:requestField>
      <otss:sequence>4</otss:sequence>
      <otss:fieldName>PAPER_COPY_REQUIRED</otss:fieldName>
      <otss:fieldValue>>true</otss:fieldValue>
    </otss:requestField>
    <otss:requestField>
      <otss:fieldName>Please provide additional comments
      here</otss:fieldName>
      <otss:fieldValue>Requiried for Government
      Grant</otss:fieldValue>
    </otss:requestField>
  </otss:serviceRequestData>
</otss:mainData>
</TSTaxpayerServiceRequest>

```

PrepareExtPaymentData for Official Payments Corporation

PrepareExtPaymentData Request

```
<ns1:TSPrepareExtPaymentData xmlns:ns1="http://oracle.com/
TSPrepareExtPaymentData.xsd">

  <ns1:head>
    <ns1:action>PREPARE</ns1:action>
    <ns1:key1>
      <ns1:name>PER_ID</ns1:name>
      <ns1:value>9351058855</ns1:value>
    </ns1:key1>
    <ns1:key2>
      <ns1:name>PER_ID</ns1:name>
      <ns1:value>2574389464</ns1:value>
    </ns1:key2>
    <ns1:webUserId>anonymous</ns1:webUserId>
    <ns1:webUserName>anonymous</ns1:webUserName>
    <ns1:emailAddress />
    <ns1:ipAddress>10.154.106.179</ns1:ipAddress>
  </ns1:head>
  <ns1:mainData>
    <ns1:paymentVendor>OPC</ns1:paymentVendor>
    <ns1:amount>33</ns1:amount>
    <ns1:paymentDate>2012-09-24</ns1:paymentDate>
    <ns1:payDestinationType>COLL_NOTICE</ns1:payDestinationType>
    <ns1:destinationDetails>
      <ns1:sequence>1</ns1:sequence>
      <ns1:fieldName>COLLECTION_NOTICE_NO</ns1:fieldName>
      <ns1:fieldValue>2574389884</ns1:fieldValue>
    </ns1:destinationDetails>
  </ns1:mainData>
</ns1:TSPrepareExtPaymentData>
```

PrepareExtPaymentData Response

```
<TSPrepareExtPaymentData xmlns:p="http://schemas.oracle.com/service/bpel/common"
xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
xmlns:ns1="http://schemas.oracle.com/service/bpel/common"
xmlns:client="http://oracle.com/TSPrepareExtPaymentData.xsd"
dateTimeTagFormat=""
xmlns="http://oracle.com/TSPrepareExtPaymentData.xsd">
  <client:head>
    <client:action>PREPARE</client:action>
    <client:key1>
      <client:name>PER_ID</client:name>
      <client:value>9351058855</client:value>
    </client:key1>
    <client:key2>
      <client:name>PER_ID</client:name>
      <client:value>2574389464</client:value>
    </client:key2>
    <client:key3 />
    <client:key4 />
    <client:key5 />
  </client:head>
```

```

<client:webUserId>anonymous</client:webUserId>
<client:webUserName>anonymous</client:webUserName>
<client:emailAddress />
<client:ipAddress>10.154.106.179</client:ipAddress>
</client:head>
<client:errorMessage />
<client:confirmationData>
  <client:header>
    <client:messageCategory />
    <client:messageParameters />
    <client:messageTxtOvrdr />
  </client:header>
</client:confirmationData>
<client:mainData>
  <client:taxpayerID>9351058855</client:taxpayerID>
  <client:onBehalfOfTaxpayerID>
2574389464</client:onBehalfOfTaxpayerID>
  <client:paymentVendor>OPC</client:paymentVendor>
  <client:amount>33</client:amount>
  <client:paymentDate>2012-09-24</client:paymentDate>
  <client:payDestinationType>
COLL_NOTICE</client:payDestinationType>
  <client:destinationDetails>
    <client:sequence>1</client:sequence>
    <client:fieldName>COLLECTION_NOTICE_NO</client:fieldName>
    <client:fieldValue>2574389884</client:fieldValue>
  </client:destinationDetails>
  <client:paymentDetails>
    <client:sequence>1</client:sequence>
    <client:fieldName>paymentAmount</client:fieldName>
    <client:fieldValue>33</client:fieldValue>
  </client:paymentDetails>
  <client:paymentDetails>
    <client:sequence>2</client:sequence>
    <client:fieldName>firstName</client:fieldName>
    <client:fieldValue>John</client:fieldValue>
  </client:paymentDetails>
  <client:paymentDetails>
    <client:sequence>3</client:sequence>
    <client:fieldName>middleName</client:fieldName>
    <client:fieldValue />
  </client:paymentDetails>
  <client:paymentDetails>
    <client:sequence>4</client:sequence>
    <client:fieldName>lastName</client:fieldName>
    <client:fieldValue>Smith</client:fieldValue>
  </client:paymentDetails>
  <client:paymentDetails>
    <client:sequence>5</client:sequence>
    <client:fieldName>suffix</client:fieldName>
    <client:fieldValue />
  </client:paymentDetails>
  <client:paymentDetails>
    <client:sequence>6</client:sequence>
    <client:fieldName>address1</client:fieldName>
    <client:fieldValue>33 Sansome St.</client:fieldValue>
  </client:paymentDetails>
  <client:paymentDetails>
    <client:sequence>7</client:sequence>
    <client:fieldName>address2</client:fieldName>
    <client:fieldValue />

```



```

</client:paymentDetails>
<client:paymentDetails>
  <client:sequence>8</client:sequence>
  <client:fieldName>cityName</client:fieldName>
  <client:fieldValue>San Francisco</client:fieldValue>
</client:paymentDetails>
<client:paymentDetails>
  <client:sequence>9</client:sequence>
  <client:fieldName>provinceCd</client:fieldName>
  <client:fieldValue>CA</client:fieldValue>
</client:paymentDetails>
<client:paymentDetails>
  <client:sequence>10</client:sequence>
  <client:fieldName>postalCd</client:fieldName>
  <client:fieldValue>94111</client:fieldValue>
</client:paymentDetails>
<client:paymentDetails>
  <client:sequence>11</client:sequence>
  <client:fieldName>email</client:fieldName>
  <client:fieldValue />
</client:paymentDetails>
<client:paymentDetails>
  <client:sequence>12</client:sequence>
  <client:fieldName>cde-Field-1</client:fieldName>
  <client:fieldValue>2574389884</client:fieldValue>
</client:paymentDetails>
<client:paymentDetails>
  <client:sequence>13</client:sequence>
  <client:fieldName>cde-Refeline-7</client:fieldName>
  <client:fieldValue>2574389884</client:fieldValue>
</client:paymentDetails>
<client:paymentDetails>
  <client:sequence>14</client:sequence>
  <client:fieldName>cde-BusiName-8</client:fieldName>
  <client:fieldValue />
</client:paymentDetails>
<client:externalID>
ddac3a78-8d0d-4589-bc8a-57306fddb69</client:externalID>
<client:feeRequirement>CNVF</client:feeRequirement>
</client:mainData>
</TSPrepareExtPaymentData>

```

Process Payment Post-back from Official Payments Corporation

PaymentPostBack from Official Payments Corporation

```

<PaymentPostBack xmlns="http://www.officialpayments.com/PaymentPostBack/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <paymentIdentifier>
f2891efc-e65d-4ca5-b9da-175586c83369</paymentIdentifier>
  <resultCode>A</resultCode>
  <resultText>Approved</resultText>
  <customDataElements>
    <customDataElement sequenceNumber="1">
COLL_NOTICE</customDataElement>
    <customDataElement sequenceNumber="2">Collection
Notice</customDataElement>
    <customDataElement sequenceNumber="3">

```

```

2574389884</customDataElement>
<customDataElement sequenceNumber="5">
f2891efc-e65d-4ca5-b9da-175586c83369</customDataElement>
<customDataElement sequenceNumber="6">
2574389884</customDataElement>
</customDataElements>
<transactionDate>2012-10-01</transactionDate>
<transactionTime>10:28:04</transactionTime>
<paymentAmounts>
  <paymentAmount sequenceNumber="1">1.00</paymentAmount>
</paymentAmounts>
<transactionFee>3.95</transactionFee>
<totalCharge>4.95</totalCharge>
<accountType>VISA</accountType>
<authorizationCode>123456</authorizationCode>
<receiptNumber>123456</receiptNumber>
<paymentID>1</paymentID>
<phoneNumber>1234567890</phoneNumber>
<name>
  <first>Amit</first>
  <middle></middle>
  <last>Seth</last>
  <suffix></suffix>
</name>
<address>
  <street1>33 Sansome St.</street1>
  <street2></street2>
  <city>San Francisco</city>
  <stateOrProvince>CA</stateOrProvince>
  <zipOrPostalCode>94111</zipOrPostalCode>
  <countryCode>US</countryCode>
</address>
<emailAddress>taxpayer@mailoption.com</emailAddress>
<paymentChannel>net</paymentChannel>
</PaymentPostBack>

```

TSOneTimePayment Request

```

<TSOneTimePayment xmlns:ns1="http://oracle.com/TSOneTimePayment.xsd">
  <ns1:head>
    <ns1:key1>
      <ns1:name>PER_ID</ns1:name>
      <ns1:value>1004567890</ns1:value>
    </ns1:key1>
    <ns1:key2>
      <ns1:name>PER_ID</ns1:name>
      <ns1:value />
    </ns1:key2>
  </ns1:head>
  <ns1:confirmationData>
    <ns1:confirmationId>EVID123456</ns1:confirmationId>
  </ns1:confirmationData>
  <ns1:mainData>
    <ns1:paymentType>APCC</ns1:paymentType>
    <ns1:payDestinationType>C1CN</ns1:payDestinationType>
    <ns1:destinationDetails>
      <ns1:sequence>1</ns1:sequence>
      <ns1:fieldValue>2574389884</ns1:fieldValue>
      <ns1:sequence>2</ns1:sequence>
    </ns1:destinationDetails>
  </ns1:mainData>
</TSOneTimePayment>

```

```

    <ns1:fieldValue />
    <ns1:sequence>3</ns1:sequence>
    <ns1:fieldValue />
    <ns1:sequence>4</ns1:sequence>
    <ns1:fieldValue />
  </ns1:destinationDetails>
  <ns1:amount>1.00</ns1:amount>
  <ns1:currency>USD</ns1:currency>
  <ns1:paymentDate>2012-10-01</ns1:paymentDate>
  <ns1:contactEmailAddress>
  taxpayer@mailoption.com</ns1:contactEmailAddress>
  <ns1:extTransactionRefID>123456</ns1:extTransactionRefID>
  <ns1:externalId>
  f2891efc-e65d-4ca5-b9da-175586c83369</ns1:externalId>
  <ns1:paymentVendor>OPC</ns1:paymentVendor>
  <ns1:externalPaymentData>
    <ns1:paymentIdentifier>
    f2891efc-e65d-4ca5-b9da-175586c83369</ns1:paymentIdentifier>
    <ns1:resultCode>A</ns1:resultCode>
    <ns1:resultText>Approved</ns1:resultText>
    <ns1:transactionTime>10:28:04</ns1:transactionTime>
    <ns1:transactionFee>3.95</ns1:transactionFee>
    <ns1:totalCharge>4.95</ns1:totalCharge>
    <ns1:accountType>VISA</ns1:accountType>
    <ns1:businessName />
    <ns1:name>
      <ns1:firstName>John</ns1:firstName>
      <ns1:middleName />
      <ns1:lastName>Smith</ns1:lastName>
      <ns1:suffix />
    </ns1:name>
    <ns1:address>
      <ns1:street1>33 Sansome St.</ns1:street1>
      <ns1:street2 />
      <ns1:city>San Francisco</ns1:city>
      <ns1:state>CA</ns1:state>
      <ns1:postal>94111</ns1:postal>
      <ns1:country>USA</ns1:country>
    </ns1:address>
    <ns1:emailAddress>taxpayer@mailoption.com</ns1:emailAddress>
    <ns1:phoneNumber>1324567890</ns1:phoneNumber>
    <ns1:recieptNumber>123456</ns1:recieptNumber>
    <ns1:paymentID>1</ns1:paymentID>
    <ns1:trafficSchool />
    <ns1:paymentChannel>net</ns1:paymentChannel>
  </ns1:externalPaymentData>
</ns1:mainData>
</TSOneTimePayment>

```

Process Payment Report from Official Payments Corporation

ReportReconciliation from Official Payments Corporation

```

<ReportReconciliation xmlns="http://www.officialpayments.com/ReportReconciliation">
  <customDataElement1>COLL_NOTICE</customDataElement1>
  <customDataElement2>Collection Notice</customDataElement2>
  <customDataElement3>2574389884</customDataElement3>

```

```

<customDataElement4></customDataElement4>
<customDataElement5>
b5e9c218-c940-4137-ad00-b9d2dac95529</customDataElement5>
<customDataElement6>2574389884</customDataElement6>
<customDataElement7></customDataElement7>
<customDataElement8></customDataElement8>
<customDataElement9></customDataElement9>
<transactionDate>20120824</transactionDate>
<transactionTime>142758</transactionTime>
<paymentAmount1>13.00</paymentAmount1>
<paymentAmount2>0.00</paymentAmount2>
<paymentAmount3>0.00</paymentAmount3>
<paymentAmount4>0.00</paymentAmount4>
<paymentAmount5>0.00</paymentAmount5>
<paymentAmount6>0.00</paymentAmount6>
<paymentAmount7>0.00</paymentAmount7>
<paymentAmount8>0.00</paymentAmount8>
<paymentAmount9>0.00</paymentAmount9>
<transactionFee>3.95</transactionFee>
<totalCharge>16.95</totalCharge>
<accountType>V</accountType>
<authorizationCode>123456</authorizationCode>
<receiptNumber>123456</receiptNumber>
<trafficSchoolFlag>0</trafficSchoolFlag>
<paymentID>1</paymentID>
<phoneNumber>1234567890</phoneNumber>
<ANI></ANI>
<firstName>John</firstName>
<middleName></middleName>
<lastName>Smith</lastName>
<street1>33 Sansome St</street1>
<street2></street2>
<city>San Francisco</city>
<reservedField></reservedField>
<stateOrProvince>CA</stateOrProvince>
<zipOrPostalCode>94111</zipOrPostalCode>
<emailAddress>taxpayer@mailoption.com</emailAddress>
<countryCode>US</countryCode>
<returnedDate></returnedDate>
<returnedDescription></returnedDescription>
<paymentChannel>NET</paymentChannel>
<uniqueIdentifier>
b5e9c218-c940-4137-ad00-b9d2dac95529</uniqueIdentifier>
</ReportReconciliation>

```

ProcessPaymentReportRecord from Official Payments Corporation

```

<TSProcessExtPayReportRecord xmlns:aia="http://www.oracle.com/XSL/Transform/java/
oracle.apps.aia.core.xpath.AIAFunctions"
xmlns:ns0="http://oracle.com/TSProcessExtPayReportRecord.xsd"
xmlns="http://oracle.com/TSProcessExtPayReportRecord.xsd">
  <ns0:head>
    <ns0:action />
    <ns0:key1>
      <ns0:name />
      <ns0:value />
    </ns0:key1>
    <ns0:key2>
      <ns0:name />

```

```

    <ns0:value />
  </ns0:key2>
  <ns0:key3>
    <ns0:name />
    <ns0:value />
  </ns0:key3>
  <ns0:key4>
    <ns0:name />
    <ns0:value />
  </ns0:key4>
  <ns0:key5>
    <ns0:name />
    <ns0:value />
  </ns0:key5>
  <ns0:webUserId />
  <ns0:webUserName />
  <ns0:emailAddress />
  <ns0:ipAddress />
</ns0:head>
<ns0:requestStatus />
<ns0:errorMessage>
  <ns0:messageCategory />
  <ns0:messageNumber />
  <ns0:messageParameters />
  <ns0:currency />
  <ns0:messageTxtOvrdr />
</ns0:errorMessage>
<ns0:paymentReportRecord>
  <ns0:paymentVendor>OPC</ns0:paymentVendor>
  <ns0:amount>13.00</ns0:amount>
  <ns0:paymentDate>2012-08-24</ns0:paymentDate>
  <ns0:payDestinationType>ClCN</ns0:payDestinationType>
  <ns0:destinationDetails>
    <ns0:sequence>1</ns0:sequence>
    <ns0:fieldName />
    <ns0:fieldValue>2574389884</ns0:fieldValue>
  </ns0:destinationDetails>
  <ns0:destinationDetails>
    <ns0:sequence>2</ns0:sequence>
    <ns0:fieldName />
    <ns0:fieldValue />
  </ns0:destinationDetails>
  <ns0:destinationDetails>
    <ns0:sequence>3</ns0:sequence>
    <ns0:fieldName />
    <ns0:fieldValue />
  </ns0:destinationDetails>
  <ns0:destinationDetails>
    <ns0:sequence>4</ns0:sequence>
    <ns0:fieldName />
    <ns0:fieldValue />
  </ns0:destinationDetails>
  <ns0:paymentType>APCC</ns0:paymentType>
  <ns0:extTransactionRefID>123456</ns0:extTransactionRefID>
  <ns0:externalId>
b5e9c218-c940-4137-ad00-b9d2dac95529</ns0:externalId>
  <ns0:externalPaymentReportData>
    <ns0:paymentIdentifier />
    <ns0:resultCode />
    <ns0:resultText />
    <ns0:transactionFee>3.95</ns0:transactionFee>

```

```
<ns0:transactionTime>14:27:58</ns0:transactionTime>
<ns0:totalCharge>16.95</ns0:totalCharge>
<ns0:accountType>V</ns0:accountType>
<ns0:businessName />
<ns0:name>
  <ns0:first>John</ns0:first>
  <ns0:middle />
  <ns0:last>Smith</ns0:last>
  <ns0:suffix />
</ns0:name>
<ns0:address>
  <ns0:street1>33 Sansome St.</ns0:street1>
  <ns0:street2></ns0:street2>
  <ns0:city>San Francisco</ns0:city>
  <ns0:stateOrProvince />
  <ns0:postal>94111</ns0:postal>
  <ns0:country>US</ns0:country>
</ns0:address>
<ns0:phoneNumber>9085478888</ns0:phoneNumber>
<ns0:emailAddress>taxpayer@mailoption.com</ns0:emailAddress>
<ns0:receiptNumber>123456</ns0:receiptNumber>
<ns0:paymentID />
<ns0:trafficSchool>0</ns0:trafficSchool>
<ns0:paymentChannel>NET</ns0:paymentChannel>
<ns0:uniqueIdentifier />
</ns0:externalPaymentReportData>
</ns0:paymentReportRecord>
</TSProcessExtPayReportRecord>
```

Appendix C

Integration with Official Payments Corporation

Official Payments XML Payment PostBack Schema

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xpp="http://
www.officialpayments.com/PaymentPostBack/" targetNamespace="http://
www.officialpayments.com/PaymentPostBack/" elementFormDefault="qualified">
  <xsd:annotation>
    <xsd:documentation>
      Official Payments Corp. Payment Post Back
      Copyright (c) 2003-2005 by Official Payments Corp. All Rights Reserved.
    </xsd:documentation>
  </xsd:annotation>

  <xsd:element name="PaymentPostBack">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="paymentIdentifier" type="xsd:string" minOccurs="1"
maxOccurs="1"/>
        <xsd:element name="resultCode" type="xpp:PaymentResultCodeType" minOccurs="1"
maxOccurs="1"/>
        <xsd:element name="resultText" type="xsd:string" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="customDataElements" type="xpp:CustomDataElementsType"
minOccurs="1" maxOccurs="1"/>
        <xsd:element name="transactionDate" type="xsd:date" minOccurs="1"
maxOccurs="1"/>
        <!-- Note that the time will include the offset from Co-Ordinated Universal
Time. -->
        <xsd:element name="transactionTime" type="xsd:time" minOccurs="1"
maxOccurs="1"/>
        <xsd:element name="paymentAmounts" type="xpp:PaymentAmountsType" minOccurs="1"
maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

```

    <xsd:element name="transactionFee" type="xsd:decimal" minOccurs="1"
maxOccurs="1"/>
    <xsd:element name="totalCharge" type="xsd:decimal" minOccurs="1" maxOccurs="1"/
>
    <xsd:element name="accountType" type="xpp:AccountTypeType" minOccurs="1"
maxOccurs="1"/>
    <xsd:element name="authorizationCode" type="xsd:string" minOccurs="0"
maxOccurs="1"/>
    <xsd:element name="receiptNumber" type="xsd:string" minOccurs="0"
maxOccurs="1"/>
    <xsd:element name="trafficSchool" type="xsd:nonNegativeInteger" minOccurs="0"
maxOccurs="1"/>
    <xsd:element name="paymentID" type="xsd:nonNegativeInteger" minOccurs="1"
maxOccurs="1"/>
    <xsd:element name="phoneNumber" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="ani" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="name" type="xpp:IndividualNameType" minOccurs="0"
maxOccurs="1"/>
    <xsd:element name="address" type="xpp:AddressType" minOccurs="0" maxOccurs="1"/
>
    <xsd:element name="emailAddress" type="xpp:EmailAddressType" minOccurs="0"
maxOccurs="1"/>
    <xsd:element name="paymentChannel" type="xpp:PaymentChannelType" minOccurs="1"
maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>
</xsd:element>

<!-- The payment result codes -->
<xsd:simpleType name="PaymentResultCodeType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="A"/>
    <!-- An approval -->
    <xsd:enumeration value="D"/>
    <!-- A decline -->
    <xsd:enumeration value="E"/>
    <!-- An error -->
  </xsd:restriction>
</xsd:simpleType>

<!-- List of Custom Data Elements -->
<xsd:complexType name="CustomDataElementsType">
  <xsd:sequence>
    <xsd:element name="customDataElement" type="xpp:CustomDataElementType"
minOccurs="1" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<!-- Single Custom Data Element -->
<xsd:complexType name="CustomDataElementType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="sequenceNumber" type="xsd:positiveInteger" use="required"/
>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<!-- List of Payment Amounts -->
<xsd:complexType name="PaymentAmountsType">
  <xsd:sequence>

```



```

<xsd:element name="paymentAmount" type="xpp:PaymentAmountType" minOccurs="1"
maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>

<!-- Single Payment Amount -->
<xsd:complexType name="PaymentAmountType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:decimal">
      <xsd:attribute name="sequenceNumber" type="xsd:positiveInteger" use="required"/
>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<!-- The account types -->
<xsd:simpleType name="AccountTypeType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="AMEX"/>
    <!-- American Express -->
    <xsd:enumeration value="DISC"/>
    <!-- Discover -->
    <xsd:enumeration value="MC"/>
    <!-- MasterCard -->
    <xsd:enumeration value="VISA"/>
    <!-- VISA -->
    <xsd:enumeration value="EC"/>
    <!-- E-Check Personal Checking Account -->
    <xsd:enumeration value="ES"/>
    <!-- E-Check Personal Savings Account -->
    <xsd:enumeration value="EBC"/>
    <!-- E-Check Business Checking Account -->
    <xsd:enumeration value="EBS"/>
    <!-- E-Check Business Savings Account -->
  </xsd:restriction>
</xsd:simpleType>

<!-- An individual's name -->
<xsd:complexType name="IndividualNameType">
  <xsd:sequence>
    <xsd:element name="first" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="middle" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="last" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <xsd:element name="suffix" type="xsd:string" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<!-- An address -->
<xsd:complexType name="AddressType">
  <xsd:sequence>
    <xsd:element name="street1" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="street2" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="city" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="county" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="stateOrProvince" type="xsd:string" minOccurs="0"
maxOccurs="1"/>
    <xsd:element name="zipOrPostalCode" type="xsd:string" minOccurs="1"
maxOccurs="1"/>
    <!-- See the following link for ISO 3166 country codes:
      http://www.iso.ch/iso/en/prods-services/iso3166ma/02iso-3166-code-lists/
list-en1.html -->

```

```

    <xsd:element name="countryCode" type="xsd:string" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<!-- An email address. -->
<xsd:simpleType name="EmailAddressType">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="([a-zA-Z0-9_\-\.]+)@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.)|((([a-zA-Z0-9\-\]+\.)+))([a-zA-Z]{2,4}|[0-9]{1,3}))(\?)"/>
  </xsd:restriction>
</xsd:simpleType>

<!-- The payment channels -->
<xsd:simpleType name="PaymentChannelType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="net"/>
    <!-- the web/internet -->
    <xsd:enumeration value="ivr"/>
    <!-- telephone through an IVR -->
    <xsd:enumeration value="filepay"/>
    <!-- a file and pay partner -->
    <xsd:enumeration value="cs"/>
    <!-- with the help of a customer service rep -->
  </xsd:restriction>
</xsd:simpleType>
</xsd:schema>

```

Official Payments Post-back XML Mapping

Official Payments Post-back XML Message			TSOneTimePayment Message			DVM Mapping
Element Name	Parent Element	Type	Element Name	Parent Element	Type	DVM/Value Source
PaymentPostBack		Outermost Tag	TSOneTimePayment		Outermost Tag	
			mainData	TSOneTimePayment Group		
			paymentVendor	mainData	Field	This value is taken from Configuration Properties
paymentAmounts	PaymentPostBack	List				
paymentAmount	paymentAmounts	Field	amount	mainData	Field	The first paymentAmount field in the list
			currency	mainData	Field	Default Currency code from Configuration Properties
transactionDate	PaymentPostBack		paymentDate	mainData	Field	Convert to xsd:Date
			bankAcctInfo	mainData	Group	
			routingNumber	bankAcctInfo	Field	
			acctNumber	bankAcctInfo	Field	
emailAddress	PaymentPostBack	Field	contactEmailAddress	mainData	Field	Same field from OPC mapped twice
customDataElements	PaymentPostBack	List				
customDataElement	customDataElements	Field	payDestinationType	mainData	Field	OTSS_PaymentDestination

Official Payments Post-back XML Message			TSOneTimePayment Message			DVM Mapping
						Populate with customDataElement with attribute sequenceNumber = 1
			destinationDetails	mainData	List	
			sequence	destinationDetails	Field	Populate the sequenceNumber from 1 to 4 in a loop
			fieldName	destinationDetails	Field	
customDataElement	customDataElements	Field	fieldValue	destinationDetails	Field	Populate the values for customDataElement in a loop where sequenceNumber = 6,7,8 and 9
authorizationCode	PaymentPostBack	Field	extTransactionRefID	mainData	Field	This is the confirmation ID coming back from OPC
customDataElement	customDataElements	Field	externalID	mainData	Field	Populate with customDataElement with attribute sequenceNumber = 5
			externalPaymentData	mainData	Group	
paymentIdentifier	PaymentPostBack	Field	paymentIdentifier	externalPaymentData	Field	Same as a unique id
resultCode	PaymentPostBack	Field	resultCode	externalPaymentData	Field	If the code is anything other than A drop the message i.e. Do not send it.
resultText	PaymentPostBack	Field	resultText	externalPaymentData	Field	New field
transactionTime	PaymentPostBack	Field	transactionTime	externalPaymentData	Field	xsd:Time
transactionFee	PaymentPostBack	Field	transactionFee	externalPaymentData	Field	
totalCharge	PaymentPostBack	Field	totalCharge	externalPaymentData	Field	
accountType	PaymentPostBack	Field	accountType	externalPaymentData	Field	
accountType	PaymentPostBack	Field	paymentType	mainData	Field	OPC_PaymentType This field is in the mainData section and will be populated using the DVM. Same field from OPC is mapped to two different fields.
name	PaymentPostBack	Group	name	externalPaymentData	Group	
first	name	Field	firstName	name	Field	
middle	name	Field	middleName	name	Field	
last	name	Field	lastName	name	Field	

Official Payments Post-back XML Message			TSSOneTimePayment Message			DVM Mapping
suffix	name	Field	suffix	name	Field	
customDataElement	customDataElements	Field	businessName	externalPaymentDataField		Populate with customDataElement with attribute sequenceNumber = 4
address	PaymentPostBack	Group	address	externalPaymentDataGroup		
street1	address	Field	street1	address	Field	
street2	address	Field	street2	address	Field	
city	address	Field	city	address	Field	
stateOrProvince	address	Field	state	address	Field	
zipOrPostalCode	address	Field	postal	address	Field	
countryCode	address	Field	country	address	Field	OTSS_Country
emailAddress	PaymentPostBack	Field	emailAddress	externalPaymentDataField		
phoneNumber	PaymentPostBack	Field	phoneNumber	externalPaymentDataField		
receiptNumber	PaymentPostBack	Field	recieptNumber	externalPaymentDataField		Same as an authorization code
paymentID	PaymentPostBack	Field	paymentID	externalPaymentDataField		
trafficSchool	PaymentPostBack	Field	trafficSchool	externalPaymentDataField		
paymentChannel	PaymentPostBack	Field	paymentChannel	externalPaymentDataField		

Official Payments Payment Report Mapping

Official Payments Payment Report File Format				TSProcessExtPayReportRecord Message DVM Mapping			
Field #	Description	Max Length	Type	Format and Example	Element Name	Parent Element Type	DVM/Value Source
					TSProcessExtPayReportRecord	OutermostTag	
					paymentReportRecord	TSProcessExtPayReportRecord	
					paymentVendor	paymentReportRecord	This value be picked from Configuration Properties
					currency	paymentReportRecord	Default the Currency code from Configuration Properties
1	Custom Data Element #1	75	Alphanumeric		payDestinationType	paymentReportRecord	OTSS_PaymentDestination Populate with customDataElement with attribute sequenceNumber = 1

Official Payments Payment Report File Format

TSPProcessExtPayReportRecord Message DVM

Mapping

2	Custom Data Element #2	75	Alphanumeric				
3	Custom Data Element #3	75	Alphanumeric				
4	Custom Data Element #4	75	Alphanumeric	businessName	externalPaymentReportData		Populate with customDataElement with attribute sequenceNumber = 4
5	Custom Data Element #5	75	Alphanumeric	externalID	paymentReportField		Populate with customDataElement with attribute sequenceNumber = 5
				destinationDetail	paymentReportRecord		Populated in a loop
				sequence	destinationDetailField		Populate the sequenceNumber from 1 to 4 in a loop
				fieldName	destinationDetailField		
6	Custom Data Element #6	75	Alphanumeric	fieldValue	destinationDetailField		Populate the values for customDataElement in a loop where sequenceNumber = 6
7	Custom Data Element #7	75	Alphanumeric	fieldValue	destinationDetailField		Populate the values for customDataElement in a loop where sequenceNumber =7
8	Custom Data Element #8	75	Alphanumeric	fieldValue	destinationDetailField		Populate the values for customDataElement in a loop where sequenceNumber =8
9	Custom Data Element #9	75	Alphanumeric	fieldValue	destinationDetailField		Populate the values for customDataElement in a loop where sequenceNumber = 9
10	Date of Transaction	8	Numeric	Format: yyyyymmdd;	paymentDate	paymentReportField	Convert to xsd:Date

Official Payments Payment Report File Format

TSPProcessExtPayReportRecord Message DVM

Mapping

				Example: 20030531			
11	Time of Transaction	6	Numeric	Format: hhmmss; Example: 134504	transactionTime	externalPaymentReportData	Convert to xsd:Time
12	Payment Amount #1	15	Currency	Format: dollars.cents; Examples: 1837.13, 11633.00	amount	paymentReportRecord	Pick the first paymentAmount field in the list
13	Payment Amount #2	15	Currency				
14	Payment Amount #3	15	Currency				
15	Payment Amount #4	15	Currency				
16	Payment Amount #5	15	Currency				
17	Payment Amount #6	15	Currency				
18	Payment Amount #7	15	Currency				
19	Payment Amount #8	15	Currency				
20	Payment Amount #9	15	Currency				
					externalPaymentReportRecord		
21	Transaction Fee	15	Currency	Format: dollars.cents; Example: 2.50	transactionFee	externalPaymentReportData	
22	Total Charge	15	Currency	Format: dollars.cents; (Sum of Payment Amount(s) and Transaction Fee)	totalCharge	externalPaymentReportData	
23	Account Type	10	Alphanumeric	See Table – Account Type, indicates credit card type	accountType	externalPaymentReportData	This field is in the paymentReportRecord section and will be populated using the DVM. Same field from OPC mapped

Official Payments Payment Report File Format

TSPProcessExtPayReportRecord Message DVM

					Mapping		
					to two different fields		
					OPC_PaymentType and maps to accountType from OPC		
					paymentType	paymentReportField	OPC_PaymentType
24	Authorization Code	50	Alphanumeric	Examples: 123456, T34997	extTransactionRef	paymentReportField	
25	Receipt Number	50	Alphanumeric	Generally the same as the Authorization Code unless Receipt Number is otherwise provided or specified.	receiptNumber	externalPaymentReportData	
26	Traffic School Flag	4	Numeric	(0=No TS; 1..9=Fee in Payment Amt. #x)	trafficSchool	externalPaymentReportData	
27	Payment ID	4	Numeric	IVR Main Menu choice or Court ID			
28	Phone Number	20	Numeric	The phone number as specified by payer	phoneNumber	externalPaymentReportData	
29	ANI	20	Numeric	IVR only - caller's phone number as collected via caller-ID			
					name	externalPaymentReportData	
30	First Name	64	Alphanumeric		firstName	name	Field
31	Middle Name	64	Alphanumeric		middleName	name	Field
32	Last Name	64	Alphanumeric		lastName	name	Field
					suffix	name	Field
					address	externalPaymentReportData	
33	Street Address #1	64	Alphanumeric		street1	address	Field
34	Street Address #2	64	Alphanumeric		street2	address	Field
35	City/Town	64	Alphanumeric		city	address	Field

Official Payments Payment Report File Format

**TSPProcessExtPayReportRecord Message DVM
Mapping**

36	Reserved	64	Alphanumeric	Reserved for future use				
37	State/Province	64	Alphanumeric		state	address	Field	
38	Zip/Postal Code	20	Alphanumeric		postal	address	Field	
39	Email Address	75	Alphanumeric		emailAddress	externalPaymentReportData		
40	Country	2	Alphanumeric	See Table - Country Code	country	address	Field	OTSS_Country
41	Returned Date	8	Alphanumeric	Format: yyyymmdd; Chargeback date				
42	Returned Description	35	Alphanumeric	Chargeback - Description ACH and NOC codes (see Return Description table)				
43	Payment Channel	10	Alphanumeric	IVR or NET	paymentChannel	externalPaymentReportData		
44	Unique Identifier	50	Alphanumeric	Used for Co-Brand+ and STP only. Unique Identifier for the transaction as provided by the Client.	uniqueIdentifier	externalPaymentReportData		Payment unique identifier

Appendix D

Setup Parameters for Official Payments Co-branding

If you choose Official Payments Corporation (OPC) as your external payment provider for the self-service application, you must contact OPC for required configuration and setup information. To co-brand an OPC page with your own information (including images), you must also complete and submit the following information.

OPC's central web site is available for viewing at <https://www.officialpayments.com/index.jsp>.

OPC Client CoBranding Parameters and Image Specifications

A. Setup CoBrand Product Information

Client Name (To be displayed):	_____
Payment Type Name (To be displayed):	_____
Batch Cut Off Time (24:00):	_____
Batch Cut Off Time Zone:	_____
Require Email Address (Y/N):	_____
Use CVV (Y/N):	_____
Display Fee Calculator (Y/N):	_____
Lock Fee Calculator Amount (Y/N):	_____

B. Setup CoBrand specific URL parameters & XML Postback Configuration

Return URL: _____

*The link that user's can click on to return to the client's site.

Redirect URL: _____

*If the user comes directly to www.officialpayments.com but is supposed to start from client site to make this payment, it will redirect the user to the client site from where the user should start the payment

Error URL: _____

*If there is an error that our system cannot handle, it will redirect the user to this URL.

Cancel URL: _____

*The link that user's can click on the Cancel button to return to the client's site.

Production Postback URL: _____

*The server our system should send authorization result information.

Must use SSL.

Test Postback URL: _____

*The server that our system should send authorization result information during testing.

Must use SSL.

Allow Multiple Payment Postbacks (Y/N): _____

Number of Postback Retries (0-5): _____

Seconds Between Postback Retries (max 5 sec): _____

C. Setup CoBrand Custom Data Elements (CDE)

*Unique Identifier is required. All other CDE's are optional.

CDE	Name	Caption	Length	Max Length	Hidden? (Y/N)
1	Unique Identifier (Required)	N/A			
2					
3					
4					
5					
6					
7					
8					
9					

D. Payment Options

American Express (Y/N): _____
MasterCard Credit (Y/N): _____
Visa Credit (Y/N): _____
Discover Credit (Y/N): _____
Visa Debit (Y/N): _____
MasterCard Debit (Y/N): _____
E-Check (Y/N): _____

E. Optional Header and Logo

Header Banner (Optional) 400w X 60h

Sample:



Logo (Optional) 64w X 64H

Sample:



Appendix E

Glossary

Term/Abbreviation	Definition
ADF	Oracle Application Development Framework.
Confirmation number	An automatically-generated unique reference identifier provided to the taxpayer on all transactions and service requests.
Casual user	Taxpayer who uses a limited set of features that do not require registration or login.
External payment provider	A third-party resource that specializes in providing automated taxation and often other types of payments, and supports automated transactional processing and reconciliation.
ITA	Interactive Tax Assistant
Managed Content	Documents that are available through WebCenter Content.
OPA	Oracle Policy Automation
OPC	Official Payments Corporation. (See "External payment provider" definition)
POI	Proof of Identity
SOA Composer (or Oracle SOA Composer)	SOA tool for managing SOA Metadata at runtime. This is a part of the SOA Suite.
Taxpayer validation	A form in which unregistered taxpayers provide basic identification, along with certain specific information pertaining to previously-recorded data, in order to access self-service application services such as payments or enquiries.
UCM	Oracle Universal Content Management, a document management tool. The UCM server is included with WebCenter installation.
WebCenter Administration Console	A part of the WebCenter application framework. It calls WebCenter Composer when you choose to add new pages or edit existing ones.
WebCenter Composer (or Oracle Composer)	WebCenter tool that allows runtime customization of WebCenter applications using the MetaData Services (MDS) in WebLogic.

Term/Abbreviation	Definition
WebCenter Content Manager (or WebCenter Content)	WebCenter Content Manager is officially WebCenter Content (formerly Oracle Enterprise Content Management). It is solution for all types of content management from file server consolidation to sophisticated multi-site web content management including UCM.
UCM Documents	Documents that are stored in UCM server.