Oracle Public Sector Revenue Management Self Service

Implementation Guide

Release 1.1.0.0

E53305-01

June 2014

ORACLE

Oracle Public Sector Revenue Management Self Service Implementation Guide

Release 1.1.0.0

E53305-01

June 2014

Documentation build: 6.22.2014 12:54:1 [TS_1403466841000]

Contents

Chapter 1

Introduction

Implementation Overview

Oracle Public Sector Revenue 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 multiple services:

Registration and Enrollment

- Provides the website user with online access to multiple tax accounts.
- Ability to request access to a category of taxes: individual, business, and/or others. Tax categories are configurable.
- Enrollment request is configurable.
  - User is prompted to enter credentials and information appropriate for the tax category.
  - UI is generated based on request definitions.
- Enrollment summary portal page displays an overview of all taxes owned by the user.
- Web service requests initiated by the enrolled user include tax account identifiers.

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).
- ITA is available to all users.

Where Is My Refund?

- Provides taxpayers with information about the current status of their refund processing.
- Dynamic, configuration-driven user interface.
- Ability to support multiple types of refund status checks.
- Response includes the expected refund amount.
- Available to all users.

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.
- Available to all users, though identification is required for casual users.

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 in real time.
  - Request can be staged for later response.
- Supports identification mode:
  - Used when user identity should be confirmed before a request can be submitted.
  - User's credentials are sent to the back-end application for confirmation.
- Available to all users, though identification may be required for casual users.

Account Information

- Provides the most important information concerning a single tax account:
  - Summary – the essential facts about account status.
  - Alerts.
  - Filing history.
  - Payment history.
- The user is offered an option to pay a tax account balance.

Taxpayer Information

- Provides the most important information concerning a single taxpayer:
  - Summary – the essential facts about the taxpayer.
  - Contact info – phones and email address.
  - Correspondence info – mailing address(es).
- The user updates phone(s) and address(es) online.

Online Form Filing

- Tax and business registration forms are filed online and submitted to the revenue management system.
  - A document locator number is assigned to the form.
  - Form data is validated before being submitted.
  - Everything that may be pre-calculated is pre-calculated.

    - Interactive and inline help is available during the filing.
    - The user may check the form data at any point.
- The user receives a confirmation email and is offered an option to pay and upload supporting documents.
- An option is available to print form data for the user's record.
- Modes of operation:
    - The form can be processed in real time.
    - The form can be staged to be processed later.
- A list of available forms is displayed on the portal page.
- Online forms are configurable:
    - Form appearance is metadata-driven. The UI is generated based on the definitions, including labels, hints, and inline help.
    - Interactive Filing advisors can be attached to form sections, specific lines, or the entire form.
    - Form availability and post-submission steps are configurable.
    - Form definition includes available actions.
    - Validation rules can be attached to the form.

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 BPM Worklist enables business users to access and act on tasks assigned to them. For example, an agent can, from a worklist, review self-service user requests.

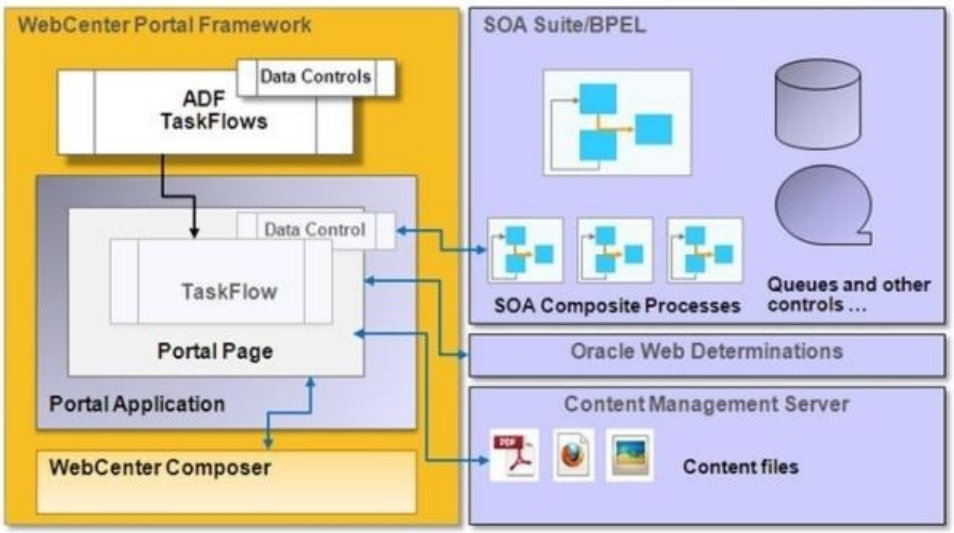
Oracle Policy Automation is a product family designed to automate rules and policies and integrate them into the customer enterprise.

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).

Oracle Business Intelligence Publisher. Oracle Business Intelligence (BI) Publisher is an enterprise reporting solution for authoring, managing, and delivering highly formatted documents, such as operational reports, electronic funds transfer documents, government PDF forms, shipping labels, checks, sales and marketing letters, and other types of publishable materials.

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 orchestrate web service processing and information exchanges between a portal application, the revenue management system, and other solution components.
- Oracle Web Determinations hosts web-based interviews used for interactive tax assistance.
- Oracle Determinations Server facilitates rulebases and provides a web service-based validation engine for online forms.
- Oracle Business Intelligence Publisher generates printable documents requested by users.

Architecture Overview

Online Services is a set of taxpayer-facing portal pages offering a wide range of self-service features, including payments and online form filing. It may serve as a foundation for a new Revenue Management Agency website or otherwise be incorporated into an existing website. The solution supports secure login and data viewing and management as well as verification of a self-service user's identity for the lifetime of a single transaction. This feature allows the casual (not logged in) user to use essential functions, such as checking the status of a refund.

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.

Online Form Validation is implemented with the **Oracle Determination Server**, which is part of the Oracle Policy Automation (OPA) product suite. This provides a high performance SOAP-based web service for fully-auditable determinations.

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 Public Sector Revenue Management Documentation section of the Oracle Technologies Network (OTN) at *http://www.oracle.com/technetwork/documentation/pubsectrevmgmt-154608.html*.

Table 1: Related Documentation and Resources

Resource	Location
Oracle Public Sector Revenue Management application documentation (If using PSRM for back-end data management.)	*http://www.oracle.com/technetwork/documentation/pubsectrevmgmt-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*
Oracle Business Intelligence Publisher Developers Guide	*http://docs.oracle.com/cd/E28280_01/bi.1111/e22259/toc.htm*
Oracle Business Process Manager Worklist	*http://docs.oracle.com/cd/E15586_01/integration.1111/e10224/bp_worklist.htm*

Chapter 2

Self-Service Application Foundation

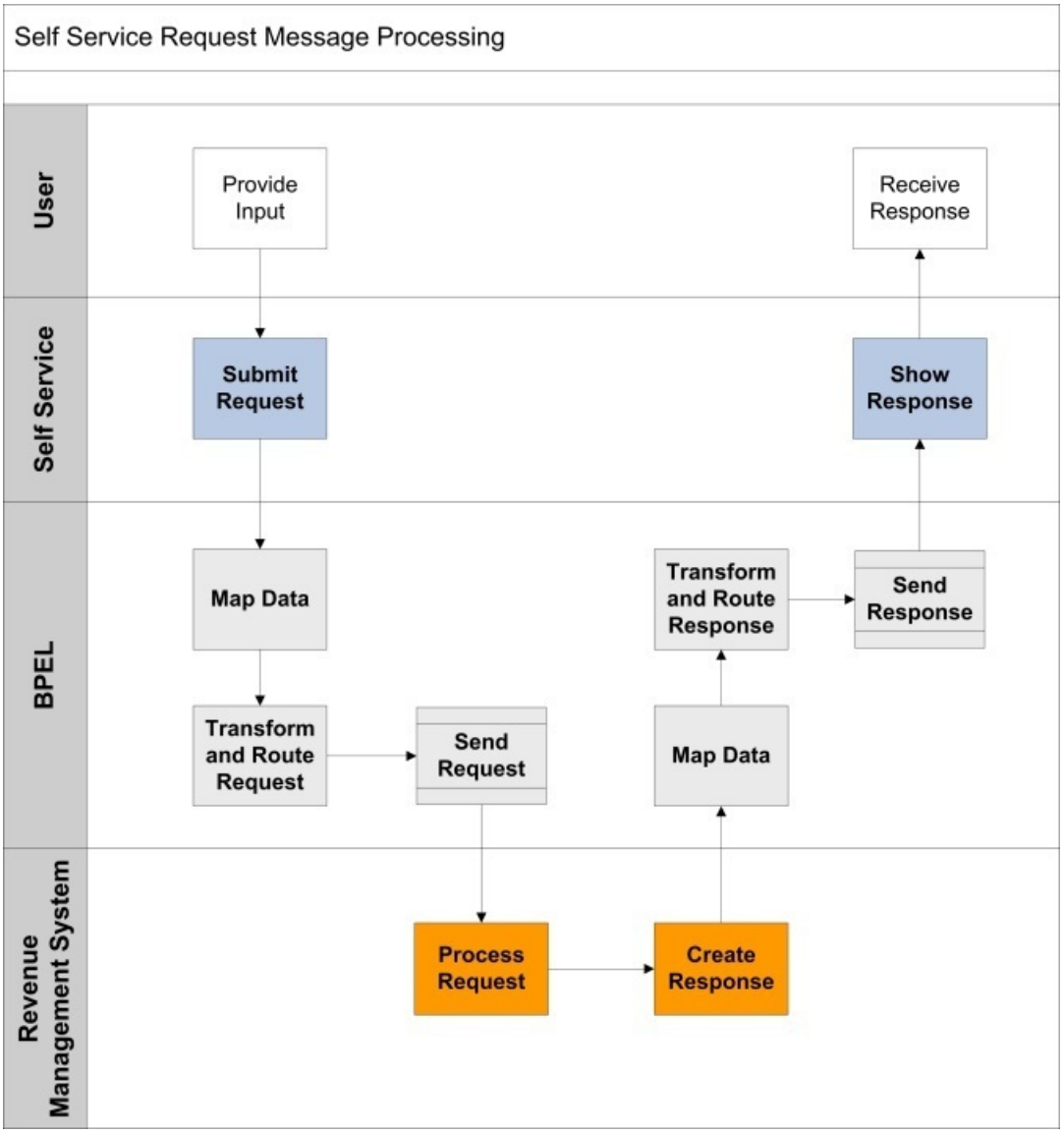
This chapter describes the processes, procedures, and design components that comprise the self-service application framework.

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.



Identification Requirements Evaluation

A person may work with self-service portal application in two ways:

- There is no need to sign in when casually viewing website content or seeking tax-related advice. Certain online services and features are available for casual users.
- User must log in and be properly authenticated and authorized in order to access sensitive personal and financial information such as historical tax returns, payment history, and other information. The web service request initiated by an enrolled user contains tax account identifiers.

The diagram below illustrates the common logic performed by online services that allow ad-hoc taxpayer identification.

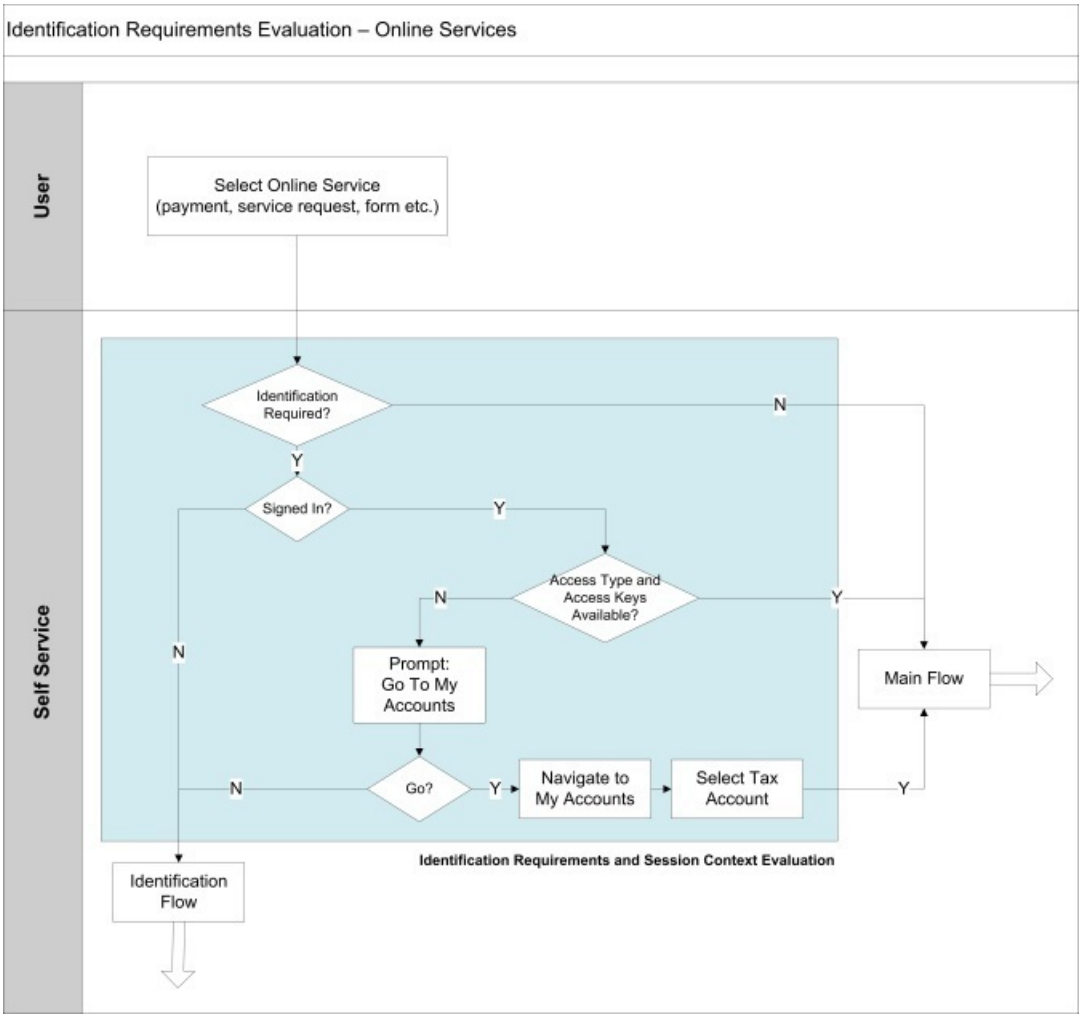


Figure 1: Identification Requirement Evaluation Flow

When the user initiates the process, the system verifies that the user has already been authenticated (logged in). If so, the system checks whether a specific tax account has already been selected. If no account has been selected, the user is prompted to navigate to the **My Account** page and make a selection. The user may choose to continue, but, without tax account identifiers in context, the user is subsequently treated as a casual user.

Possible scenarios include:

User Registered	Enrolled/Account Selected	Identification Required	Result
Yes	No	Both	Identification Invoked
	Yes	Both	Identification Invoked
Yes	No	Unregistered	Identification Invoked
Yes	Yes	Unregistered	Main Request Invoked
No	N/A	Both	Identification Invoked
No	N/A	Unregistered	Identification Invoked
No	N/A	None	Main Request Invoked

Tax Account Access

When the tax authority grants user an access to his/her taxes, an Access type describes the scope of a single tax account managed via self service portal.

It is referenced in various self service-related configurations; when the request is processed the revenue management system may access type to determine an appropriate logic to execute. For example an outstanding balance for a specific tax is retrieved in a different way than the overall outstanding balance for a taxpayer.

Access Keys

The tax account is identified by the access type and a set of keys. The integration supports an ability to identify with up to 10 keys.

The first key is always reserved for the taxpayer ID associated with the tax account.

Line of Business

In web self-service application Line of Business is used to broadly group areas of interest. The web pages found within a defined line of business are related to the information and tasks appropriate for that area of interest. Some example of lines of business for an implementation may be Individual taxpayers, Business taxpayers, and Tax Preparers. Another implementation may use even more granularity. For example, Non-profits may be another line of business.

Session Context

Session context is the information available to the portal application components for the duration of the single authenticated browsing session. Upon successful login, the self service user is expected to choose a tax account to view and manage. Session context is populated with user's details (username and email, if available) and selected account's identifiers (access type, access keys and line of business).

The title and the taxpayer name of the tax account in context are displayed on the top right corner of the main page area.

Session Context Evaluation

The system evaluates the session context in order to let user access the tax account-related information.

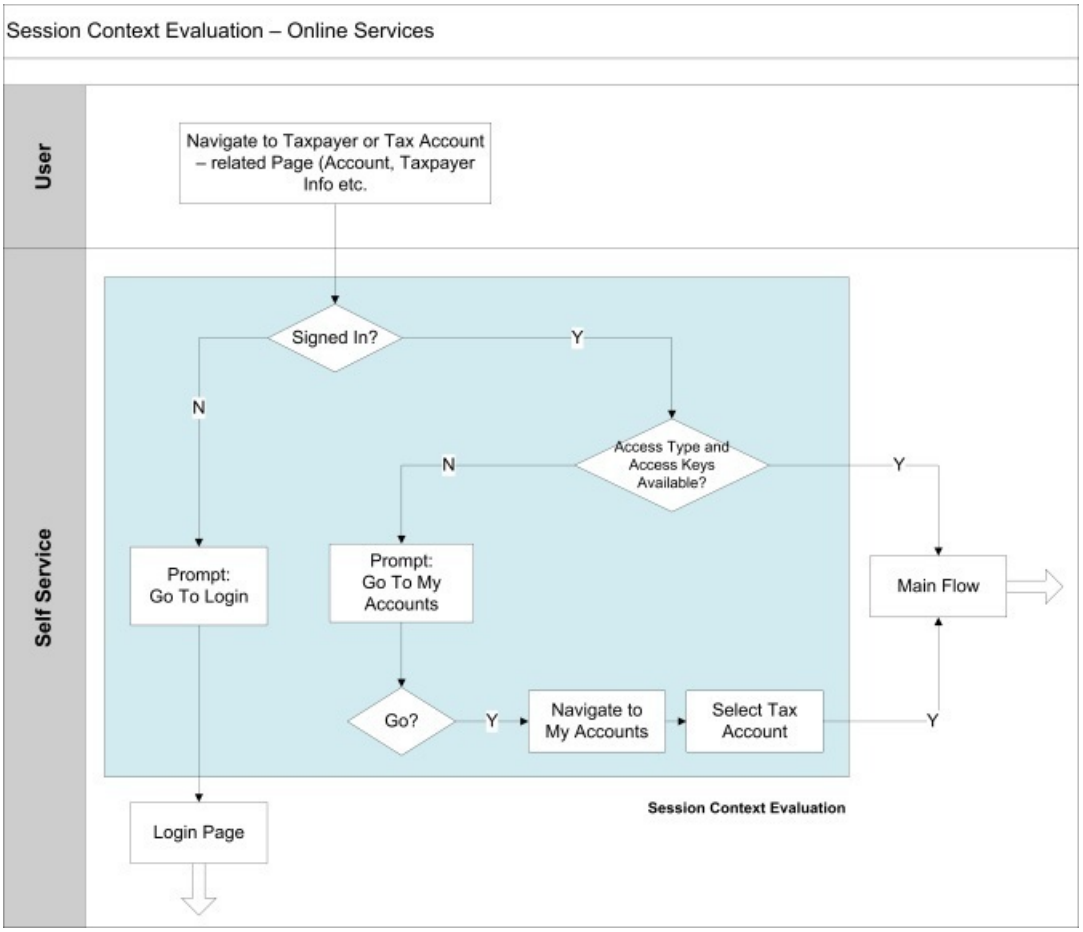


Figure 2: Session Context Evaluation Flow

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 10 name/value pairs for business identifiers whose combination represents a single tax account.

  Note:

  - The first key is reserved for the unique taxpayer identifier.
  - Supported Key Name and Value format: 30 characters, alphanumeric.

- **Access Type** – Alphanumeric code defining a type of Access Keys combination.
- **Error** – Reserved for the business error messages returned by the revenue management system.
- **Confirmation** – Contains the confirmation ID and message(s).

Additional optional elements included in multiple messages:

- Action indicates what type of special processing is requested.
- Linked request element contains the confirmation number of the related web service request.
- Line of business.

For additional information, see *Common Features of All BPEL Processes* and "Common Components" in the *WSDL Library (Appendix A)*.

Transaction Confirmation

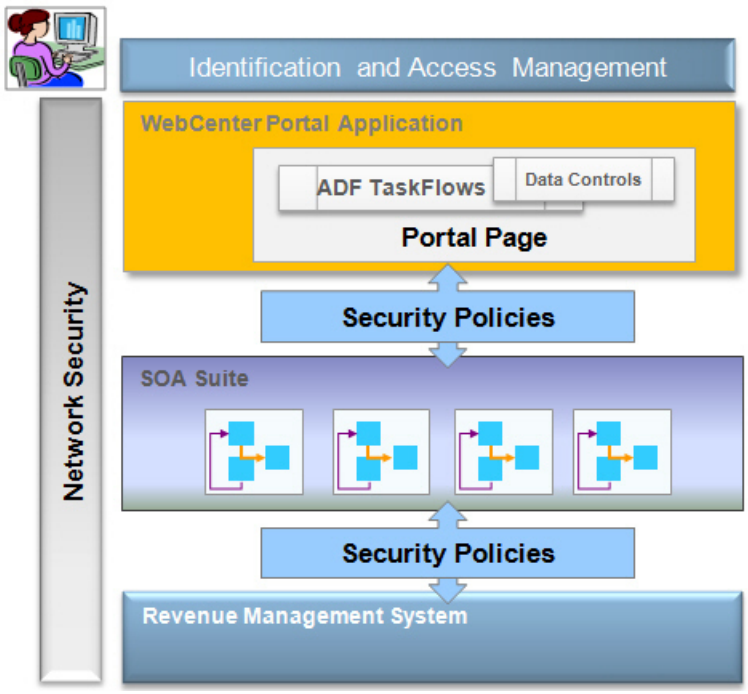
Confirmation numbers are generated in the integration layer and assigned to the <confirmation Id> element. See *Common Features of All BPEL Processes* and *Confirmation Number Utility Service* for details.

Application Security

The base product portal application includes the following page categories:

- **Secured pages - Admin**. These pages are used for administration purposes and are typically accessed by administrators or implementers.
- **Public pages**. These pages provide unauthenticated users with access to a limited set of self-service features and website content.
- **Secured pages – Self-Service User**. These pages provide authenticated users with access to their tax accounts.

Portal Application Security



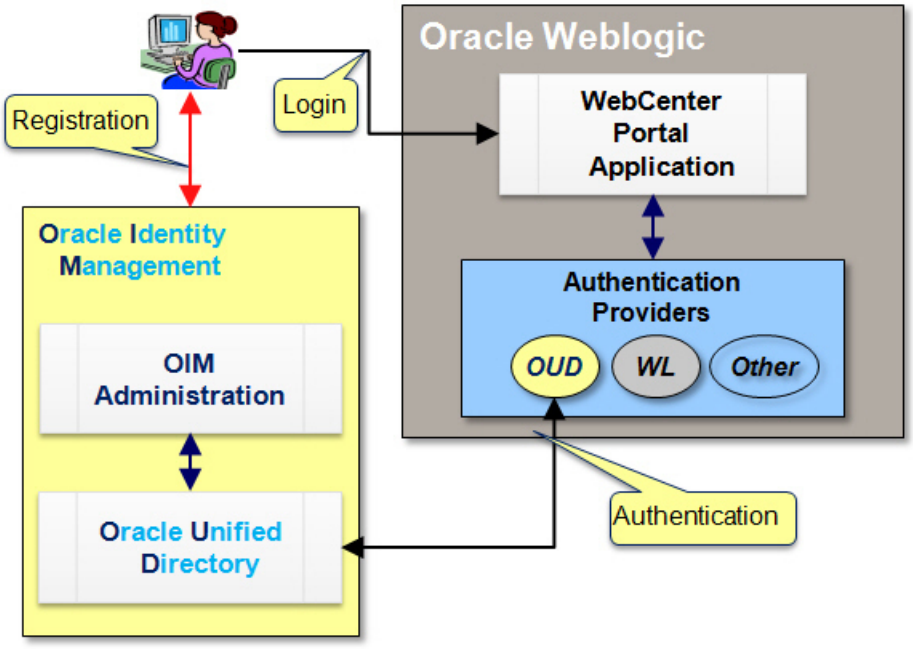
The portal application uses the WebCenter security model, integrated with WebLogic to enforce user authentication. The self service product security model includes multiple layers:

- The Network Security layer defines the access to network nodes for specific users, groups, roles, etc. The specific setup and deployment of self service product-related hardware and software is expected to be defined by customers

according to their unique security requirements. Such requirements can include firewall settings, database security, system administration rights, etc.

- The Identification and Access Management layer defines the access right of individuals or groups to the system. This includes administration and end-user rights. Self service product identification and access management is pre-integrated with the Oracle Identity Management suite of products:
  - Oracle Identity Management for user registration and authentication.
  - Oracle Unified Directory connects Oracle Identity Management to the WebLogic.

  The implementation may use the Oracle Unified Directory in conjunction with alternative security/authentication providers.



- The application security layer is responsible for:
  - Access rights of individuals or groups to various resources on the portal are provided by the WebCenter portal application framework.
  - Access rights for the specific tax account data is enforced by the application itself. Users can only see information related to specific accounts determined by an enrollment process. The enrollment process identifies the portal users as tax account owners (by a matching a configured set of credentials) and allows them to access their account information.
- The Web Services Security layer defines the communication standards between the portal application, BPEL and back-end system. Web Services security policies can be defined for internal message exchange between all product layers. Supported policies include Simple HTTP tokens, SSL, SAML. These security settings are typically configured using Oracle Enterprise Manager.

The base product portal application includes the following application roles:

- Default WebCenter roles and groups:
  - **Administrator** – Mapped to the WebLogic Administrators group. Any WebLogic user that is linked to that group will have the Administrator role in the application.
  - **AppConnectionManager** – Not used by the portal application.
  - **AppConnectionViewer** – Not used by the portal application.

- **PSRMSSFormsAdmin** – An enterprise application role that is essentially a WebLogic group. Users assigned to this group will have access to secure application features associated with the portal forms administration, including locking and unlocking, retiring and re-importing the form definition.

The user ID and initial password for the **Administrator** role are provided as part of the installation procedures and may be modified later.

Additional user group(s) and roles should be defined for the actual self service customers whose access should be limited to the online services (self service secured) and public.

If your implementation wishes to use Oracle Identity and Access Management, see the *PSRMSS Installation Guide* for installation and configuration instructions.

Portal Application Admin Pages

All portal pages under the **Settings** and **Configurations** menus are initially 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 admin pages access.

Portal Application Public Pages

Public pages are available to all users.

Some of the portal's functional pages, such as online payments or refund status requests, as well as an online services "entrance" page, are all public pages.

The portal home page for taxpayers is:

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

Portal Application Self Service Pages

Only authenticated website users who successfully log in can access these pages. These pages provide taxpayer with ability to view and manage tax account data. The information displayed on these pages is retrieved based on the current session context which includes tax account identifiers and related taxpayer ID.

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.
- Interactive tax assistance.

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 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.

- **Business error**. A web service request sent to the back-end system for processing resulted in a valid business error. The response is expected to contain the error message number and parameters.
- **System availability**. The message, "Back-end system is not available" means the system is down or not responding.
- **SOAP fault**. The message, "Web service call result from back-end system is a SOAP fault" is typically the result of a system error in the back-end system.

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

Messages communicate information in response to a portal user's actions, including confirmation data, inquiry results, and errors. Errors originate from various sources:

- ***Revenue Management and other Back-End Systems*** - The result of a web service request sent as part of the service supported by the base product (e.g., a payment request).
- ***Middleware (BPEL)*** - Translation of a message originating in the revenue management back-end system or a message created in this layer due to processing exceptions.
- ***Portal Application*** - An error thrown by the portal application while processing a user request.

Revenue Management 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 reason for the failure 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.

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 throws errors or displays informational messages under the following circumstances:

- **Instant validation of the user's input**: This may happen during both admin data maintenance (system settings or configurations) and functional transaction interactions (e.g., input for a refund status request).
- A web service response in error (e.g., making a payment or inquiring about refund status).
- System error messages as a result of a failure to perform or complete an operation. Examples:
  - 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.

A special category of messages may be defined in the portal application for displaying formatted and localized information on screen.

Displaying Messages

A standard XML fragment for the message includes:

- **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

  .
  A message source indicator (for example, REM for revenue management, BPEL for Integration Layer), and an original Message Category and Number are included in the response XML and displayed to the website user.
- **Message Number** – A number or a string.
- **Message Parameters**:
  - **Parameter Sequence** – A number.
  - **Parameter Value**.

Override Message Text – The web service response may contain a fully-formatted message in text or HTML format.

The following logic is used when a message is returned from a web service call:

- If the override message text is returned, it is displayed.
- 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 are suppressed. A message is considered a 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 and is based on the message parameter data types defined in Portal Application. Substitution variables with no value are replaced by **blanks**. Parameter values with no corresponding substitution variables are ignored.

How Messages and Parameters are used in the Application

Error or Confirmation Messages: The message number and parameters are retrieved as part of the web service response. The portal application reads the message definition and composes text, substituting the parameters according to message definitions.

Application Information Messages: A set of parameters is retrieved as part of the web service response. The portal application reads the configuration, retrieves the message number to be used, and composes the display text, substituting the parameters according to message definitions.

Portal Message 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 must be restarted.

Portal Application

Portal Application Overview

The base product portal application comprises a set of functional admin pages. The portal application also includes a special **Login Page**.

All base product pages are delivered referencing the default portal application template. The application includes a single page template; thus, all pages initially share a common structure.

Login Page

Two links-- ***Forgot your Password?*** and ***Sign Up*** --point the user to the registration/identity management location, where the user may establish or restore a secure login.

The following elements on the page are configurable:

- The target URLs for both links are defined using the system options **USER_FORGOT_PASSWORD_URL** and **USER_REG_URL**. For additional details on these options, see *Defining System Options*.
- The link text can be modified through the override bundle, in the entries **TS_LOGIN_FORGOT_PWD_LBL** and **TS_LOGIN_SIGNUP_LBL**. For more details, see *Static Text Multi-Language Support*.

The contents of the page may be customized using WebCenter Composer.

Portal Page Template

The portal page template includes the following sections:

The top ("header") region includes:

- The top bar image and company logo placeholder.
- Current User ID information, a link to the WebCenter Administration Console (for customizations), along with login and language selection links.
- The main navigation bar. This bar displays the entire default portal navigation model.

- A "breadcrumbs" region immediately below the main navigation bar displays the recent sequence of portal pages visited by the user.
- The name and title of the tax account currently in context and the hyperlink Switch Account that provides navigation to the **My Accounts** page.

  Note: The availability of the navigation options on a specific page depends on the security settings; only those pages that the self-service user is allowed to access are visible.

The main page area below the main navigation bar is used for the page content, which varies for each specific page.

The bottom section includes:

- A custom icons bar that can be configured to show icons/links to social media sites such as Facebook, and/or other external sites.
- A custom links bar that can be configured to show links to common portal 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 maintain basic definitions that are used across the system.

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 navigation panel.
- The middle part of the page is a placeholder for customer-specific HTML content.

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 structure similar to the admin "entry" pages:

- The top and bottom sections of the page are taken from the page template.
- The left side navigation panel.
- The middle part of the page is a place holder for customer-specific HTML content.

The functional pages have the same structure as the admin pages. The area in the middle of the page contains task flow(s) related to a specific functionality, e.g., 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 *Customizing the 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).

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.

  Note: The date format is defined in the *Extended Value* of the lookup definition. The extended value is the actual format that is used by the system.

- **DEF_DATETIME_FORMAT** defines the DateTime format that the portal application will use when communicating with the revenue management system. The valid values for this option are stored in the **DATETIME_FORMAT** Lookup.

  Note: 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_BACKEND_DATETIME_FORMAT** defines the DateTime format that the portal application will use when communicating with the revenue management system. The valid values for this option are stored in **DATETIME_FORMAT** Lookup.
- **DEF_BACKEND_TIME_FORMAT** defines the Time format that the portal application will use when communicating with the revenue management system. The valid values for this option are stored in **TIME_FORMAT** Lookup.
- **DEF_COUNTRY** defines the Country the portal application will use when displaying Address information. The valid values for this option are stored in **COUNTRY** Lookup. The default value delivered is USA, additional values can be defined as needed.
- **DEF_EMAIL_CONFIG** defines default Email Configuration to be used for confirmation and service request emails.
- **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.
- **DEF_SYS_BOOKMARKS** references the list of base portal application pages. The values are stored in **SYS_BOOKMARKS** Lookup.

- **DEF_TAXPAYER_ACCESS_TYPE** defines the access type used by default for enrollment and account information management. The default value, **TAXROLE**, can be modified according to the business requirements.
- **DEF_FORM_CONFIRM_MSG** defines the default message to be used for the form submission confirmation. Default value is 51301.
- **BOOLEAN_LOOKUP** defines the Lookup whose values provide translation for boolean message parameters. Review the values of the base lookup **BOOLEAN_PARAM** and customize it according to your business needs.
- **UPL_FILE_TYPE** defines file types that the self-service user is allowed to upload. File types are delimited by comma. The default value, **pdf**, can be modified according to business requirements.
- **USR_REG_URL** defines the location of an Identity Management site registration page where non-authenticated users are redirected via the **Sign Up** link.
- **USR_FORGOT_PASSWORD_URL** defines the Identity Management web page location where the user is redirected via the **Forgot Password** link.

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 uses a free-form text value, so no validation is performed.
- **MAIL_SMTP_PORT** defines the port number of the SMTP server. This option uses a free-form text value, so no validation is performed.

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 in this field. For example, the code "en" is for English and should only be defined once.

Extended Locale is a corresponding country-specific four-letter locale.

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 a manual refresh. The corresponding action button is provided on the Message Search page.

Messages that are defined in the portal application are used as follows:

- 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. Messages are used to communicate errors and information to the user, and also to display various formatted data on-screen.
- 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.
- Messages store text fragments used by the OPA Rulebase Model generator.

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.

Use the **Refresh Cache** button to update application cache and make all recent message modifications available.

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 describes the general purpose of the message and may also indicate the message's origins. Valid values for the category are:

- **Self Service - Information**. Messages of this category are used for portal application information and warnings. An example of a message from this category is: "Navigate to My Account page and select an account".
- **Self-Service – Error**. Messages of this category are used to report portal application errors. An example of a message from this category is: "The input date format is invalid".
- **Revenue Management**. Information messages of this category are used for non-error responses from the back-end system, including various confirmation messages.
- **Revenue Management**. Error messages of this category are used for validation errors received from the back-end system.
- **BPEL – Information**. Messages of this category are used for non-error responses from the integration layer.
- **BPEL – Error**. Messages of this category are used for errors occurred in the integration layer.
- **Application Information**. Messages of this category are used to display formatted data on-screen on, for example, the **Account Summary**, **Alerts**, and **Enrollment Summary** pages.
- **OPA Data Model Info**. This is a special group of messages used internally by the OPA Rulebase Model generator for rulebase entities and attribute text.

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. Message text may contain basic HTML tags controlling display attributes such as line break, font weight etc.

The **Message Parameters** collection defines data types for the message's substitution parameters. Supported parameter value data types are defined in the lookup **MSGPARMTYPE**. The values are translated based on the parameter type. The value of the parameter whose data type is **Lookup** is substituted using the description of the corresponding **Lookup Value**.

Example: A message is to be defined as follows:

- **Message Number:** 11111
- **Text:** "The {1} payment of {2} received on {3} will be posted to your account."
- Parameters:

- `1 – Type Lookup TENDERTYPE`
- `2 – Type Currency`
- `3 - Type Date`

Assuming the actual XML message data contains message number 11111 and three parameters with values `CASH`, `33`, and `01-01-2001`.

At runtime, the system merges the values of the three parameters merged into the message before it is displayed to the user. The resulting message would be displayed as:

```
"The cash payment of $33.00 received on 01/01/2001 will be posted to your account."
```

> **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 **Add**, **Edit** and **Delete** functions are available for each lookup definition.

You can click the **Duplicate** button to copy a selected lookup.

Lookup Maintenance

Lookup Code is the unique identifier for the lookup.

The **Owner** indicates whether the message definition is owned by the base product or by the implementation. The system sets the owner to *Customer Modification* when an implementer defines a new message. This information is display-only. The system sets owner to *Revenue Management* when the Lookup is imported as part of Form Definition import.

Use **Override Description** to change the description of a base product lookup.

The **Customizable** checkbox indicates whether or not new values can be added to the lookup. This is mostly applicable to base product lookups.

Help text can be added to any lookup. 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.

Use **Override Help** to change the help text that displays for 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.

Click the **Add** button to define a new lookup value.

The **Reload Values** button is visible if the Lookup's owner is Revenue Management. When this button is clicked the system invokes the web service *Refresh Lookup*. This service returns a list of values from the back-end system. New values are added in **Active** status. Values that are no longer on the list are deactivated.

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 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** specifies the graphic associated with the lookup value. This value is used by the system under special circumstances, e.g., to illustrate the payment status.

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 Validation Rules

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

Use **Settings > Validation Rule** to reach **Validation Rule Search**.

Validation Rule Search

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.

Rule Type defines validation mechanism and usage. Supported options are:

- Regular Expression rules are used to validate a single field value.
- OPA Rulebase rules are used to validate the entire object data via web service.

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.

The **OPA Rule** references an Interview (rulebase) invoked to validate the object data. The rule must belong to the Interview Set whose category is ***Data Validation*** .

Defining Property Type

Custom Properties is a name-value collection existing on multiple entities, e.g., on a Form, Service Request, etc. Custom Properties are searchable and can be copied to the web service request. They can be used by the implementation as a means to deliver additional information to the back-end system and/or to group configuration items.

The **Property Type** entity defines the attributes of a Custom Property.

Property Type Search

The user can search existing property type definitions.

The standard **Edit** and **Delete** functions are available for each property type.

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

Property Type Maintenance

Property Name and **Description** appear on the search and dropdowns.

Override Description can be used by the implementation to replace the product-supplied description.

Value Type defines the type of data that could be used as property value. Supported options are *Free-form Text* and *Lookup*.

Lookup should be specified if the **Value Type** is *Lookup*. At runtime, the appropriate dropdown is displayed on the Custom Properties grid.

Unique indicates if the entity may have more than one property of this type.

Allow Search indicates if this Property could be used as search criteria on the "owner" entity's main search.

Detailed Description is used internally to capture detailed explanation about the property.

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.

Common Configurations

Address Configuration

Web services used by a portal application include a common XML fragment for address data:

```
<location>
<addressId/>
<addressUsage/>
<effectiveDates>
<startDate/>
<endDate/>
<seasonStart/>
<seasonEnd/>
</effectiveDates>
<address>
<name/>
<country>
<address1/>
<address2/>
<address3/>
<address4/>
<streetNumber1/>
<streetNumber2/>
<county/>
<city/>
<state/>
<postal/>
<houseType/>
<latitude/>
<longitude/>
<inCityLimit/>
<comments/>
</address>
```

```
<overrideFormattedAddress/>
</location>
```

The non-customizable Lookup ***ADDRESSFIELD*** contains the list of all available elements of the fragment.

Address Configuration defines the subset of standard address elements that is applicable for the country. It also controls how the portal displays formatted address information.

Use the **Configurations** menu to navigate to **Address Configuration**. You may search for an existing record or use the **Add** button to create a new entry.

Prerequisites for adding new address configuration:

- Country codes are defined using the Lookup **COUNTRY**. Before adding new address configuration make sure that the country you wish to configure has a corresponding active value in the Lookup COUNTRY.
- Define an application message that will be used to compose the formatted address information. Address fields are used as substitution parameters for this message. The base product provides message 40010 for this purpose.

Address Configuration definitions include:

- **Country** – Select a country from the drop-down list. Only one configuration per country code is allowed.
- **Formatted Address Message** - Message used to compose a formatted info. Address field values are used as substitution parameters for this message.
- **Address Fields List** - List of all address XML elements applicable to the Country's addresses. The list should contain all address elements applicable for the country, even though not all of them might be included in the formatted info. The following attributes should be configured for each element:
  - Address Element and Description referencing an XML element. The selection is based on values in the **ADDRESSFIELD** lookup.
  - **Reference Field** - Field whose definitions are used for the element when Address is rendered on the UI. It includes label, data type, and possibly the Validation Rule linked to the Field.

    Note: Country must be always included in the address fields list.

    Examples:

    For Country *United States*, element **<state>** may reference a Field STATE, with label ***State*** associated with Lookup ***STATE*** whose values contain US states.

    Address configuration for the country *Canada* for element **<state>** would reference a Field PROVINCE, with label ***Province*** associated with Lookup ***PROVINCE*** whose values contain Canadian provinces.
  - **Visible** - Indicates if the field should be visible on screen in display-only mode.
  - **Required** - Indicates if the element is required.
  - **Display Sequence** - Defines the position of the element on-screen.
  - **Formatted Info Parameter** - Maps the element's value to the substitution parameter within the formatted **AddressMessage**.

Access Type

Access type describes the scope of a single tax account managed via the self service portal.

Use the **Configurations** menu to navigate to the **Access Type** configuration. You may search for an existing record or use the **Add** button to create a new entry.

The **Access Type** is defined as follows:

- **Access Type** and **Description** are standard attributes for admin configuration.

- **Detailed Description** is for internal use.
- **Pay Current Balance** - Payment Destination is used when the self-service user chooses to pay the tax account balance. When the one-time payment flow is triggered, the payment destination details collection is populated with access keys in context.

  Note: Payment Destination **TAX_ROLE** is provided. It has one field and expects a single key identifier of the tax account.

For each supported **Line of Business**, the Access Type defines:

- **Account Summary Title Message** - This message is used to compose the Title. The *Get Account Summary* web service response contains **Account Summary Title** parameters. They will be used as Message substitution parameters.
- **Account Summary Details Message** - This message is used to compose the main text of the summary. The *Get Account Summary* web service response contains Account Summary Details parameters. They will be used as Message substitution parameters.
- **Enrollment Summary Title Message** - This message is used to compose the Title of the enrollment summary.
- **Enrollment Summary Details Message** - This message is used to compose the main text of the enrollment summary.

When defining the Access Type, make sure that the Summary Title and Details parameters data types are coordinated with the corresponding Message's parameters. If the parameter's data type is ***Lookup*** , the values returned by the web service should match one of the Lookup's values.

Example

Assuming your Account Summary Title message should read: `ABC Corporation, Ltd - Corporate Income`

The Message text '`{1} - {2}`' has two substitution parameters:

- 1 - data type ***Text***
- 2 - data type ***Lookup*** , Lookup ***TAXTYPE***

In order to display the Account Summary Title properly, the summary title parameter value CORP-INCOME (Corporate Income) returned by the web service should exist within values of the Lookup ***TAXTYPE.***

Design Account or Enrollment Summary for a New Access Type

When designing a new Access Type, determine what information the web service needs to retrieve. It may include amounts, dates and other important data.

Create messages to be used as the Enrollment and Account Summary Title and Details.

Create new Lookups and populate them with values matching those in the Revenue Management system.

Email Definition

Email Definition defines templates for emails sent by the system. The configuration is language-sensitive and can be set up in all supported languages. The ***RECIPIENT*** lookup stores the set of pre-defined agency-owned email addresses.

Navigate to **Configurations > Email Definition**. Email definitions are configured as follows:

- Select **Configurations > Email Definitions** to maintain email definitions.
- **Active** indicates that an email definition is available for use.
- **Allow Attachment** indicates if an attachment can be added to this email.
- Use **Sender** to specify a ***From*** email address. The self-service user will receive an email from the sender.

- Use **Recipient** to specify an email address of an entity that will receive the email, for example, `formprocessing@myCompany.com`.
- Use **CC** to specify an additional email address of the department or customer support group that is supposed to receive a copy of the outgoing email.
- Use **BCC** to specify a hidden recipient (e.g., an audit/backup email box).
- The text entered in **Subject** will appear as the email subject.
- Use **Body** to define a static portion of the email body.
  - Use *{message_details}* notation to mark the position of a dynamic message text that will be injected into email body. HTML and plain text are supported.
  - Special use for email-based service requests: Enter *{request_details}* notation to mark the position at which the system should inject the service request details.

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.

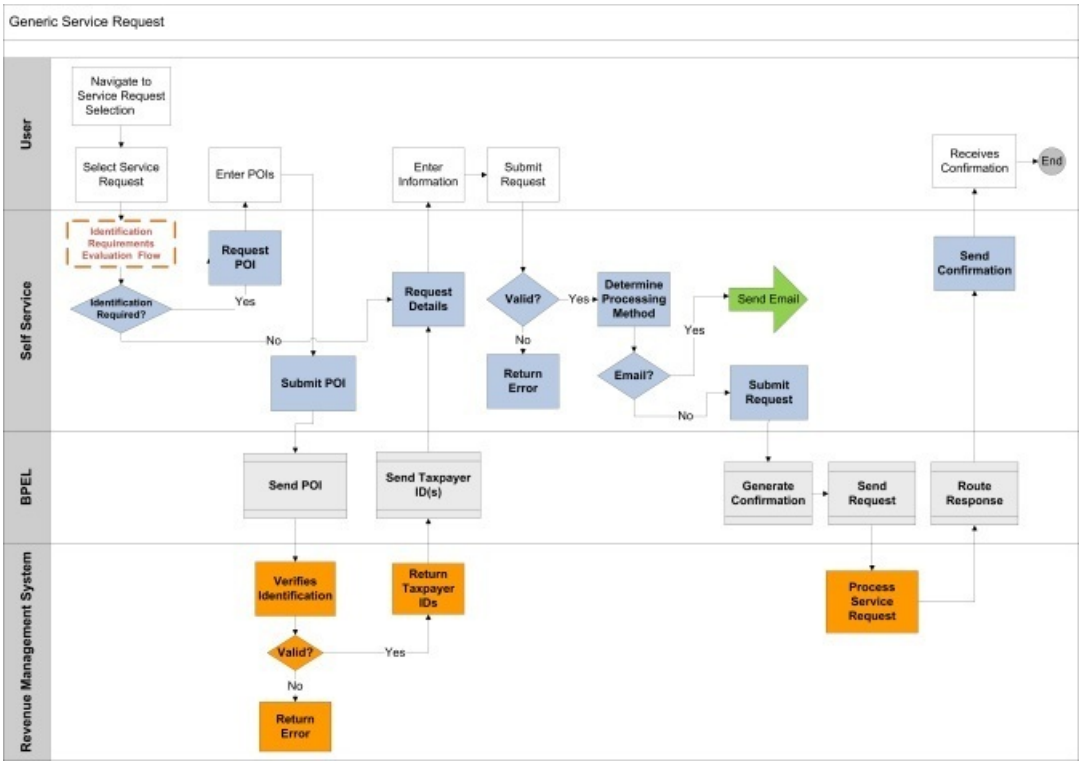
Several special categories of service requests are used by other application features. These features use the Service Request infrastructure for dynamic UI rendering and configurable user input. However, the request processing and the response handling are different from the regular service request. Examples of such special categories are **Refund Status Inquiry Request**, **Enrollment Request**, and others.

Regular 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 drop-down list. Each topic corresponds to an active Service Request.

The service request may be available to all website users or offered exclusively to registered and signed-in users. This restriction is controlled by service request configuration.

After a topic is selected the system reads the Service Request configuration. It first determines whether this service request requires taxpayer identification.

For additional information, see *Request/Response Message Flow*.

Identification

If identification is required, 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.

Submit the Request

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

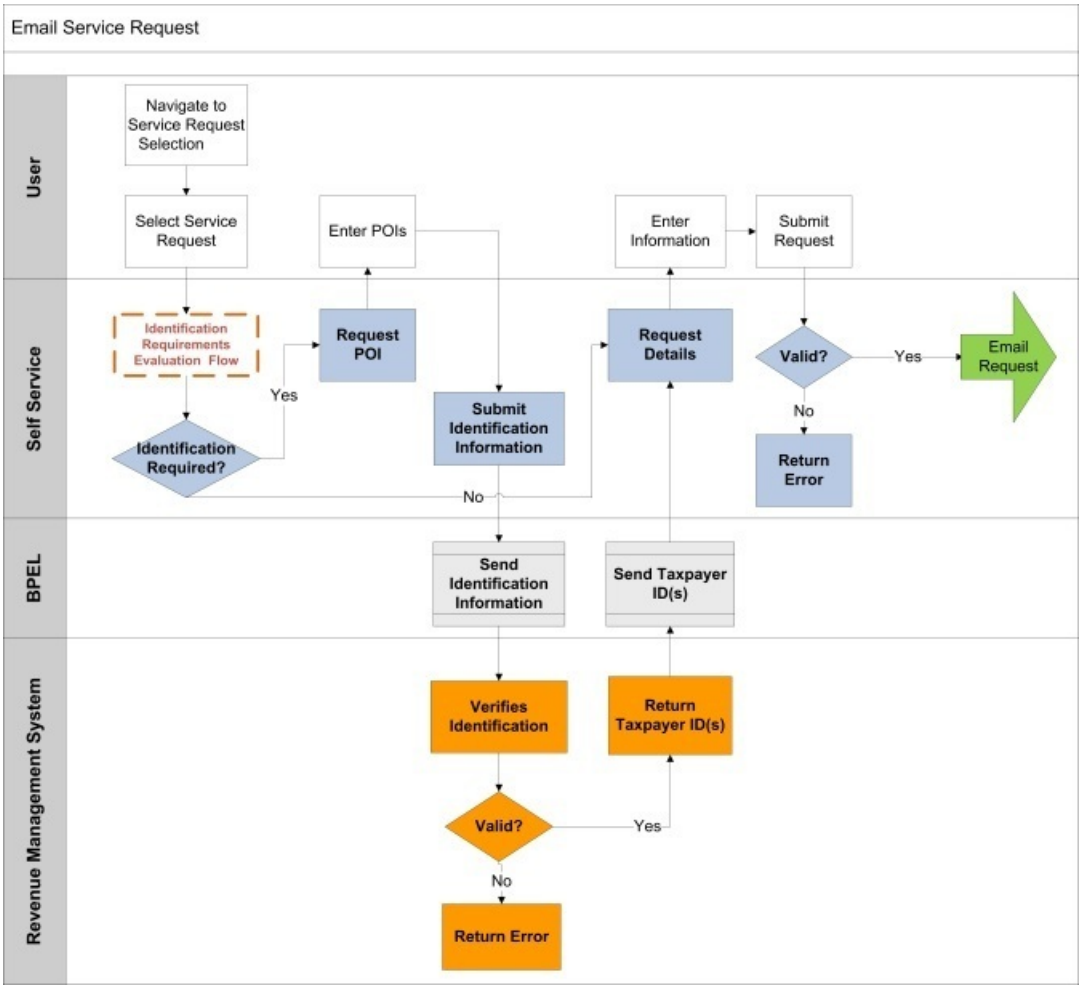
If the web service call was successful, the response contains the confirmation details.

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.

Email 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.

For additional information, see *Request/Response Message Flow*.

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* section for more details.

Email the Request

The target ("From") email address is taken from taxpayers 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 using the email text derived from the email definition referenced on the service request. The data entered by the user is injected into a placeholder position within the email text from the list of the service request fields. 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.

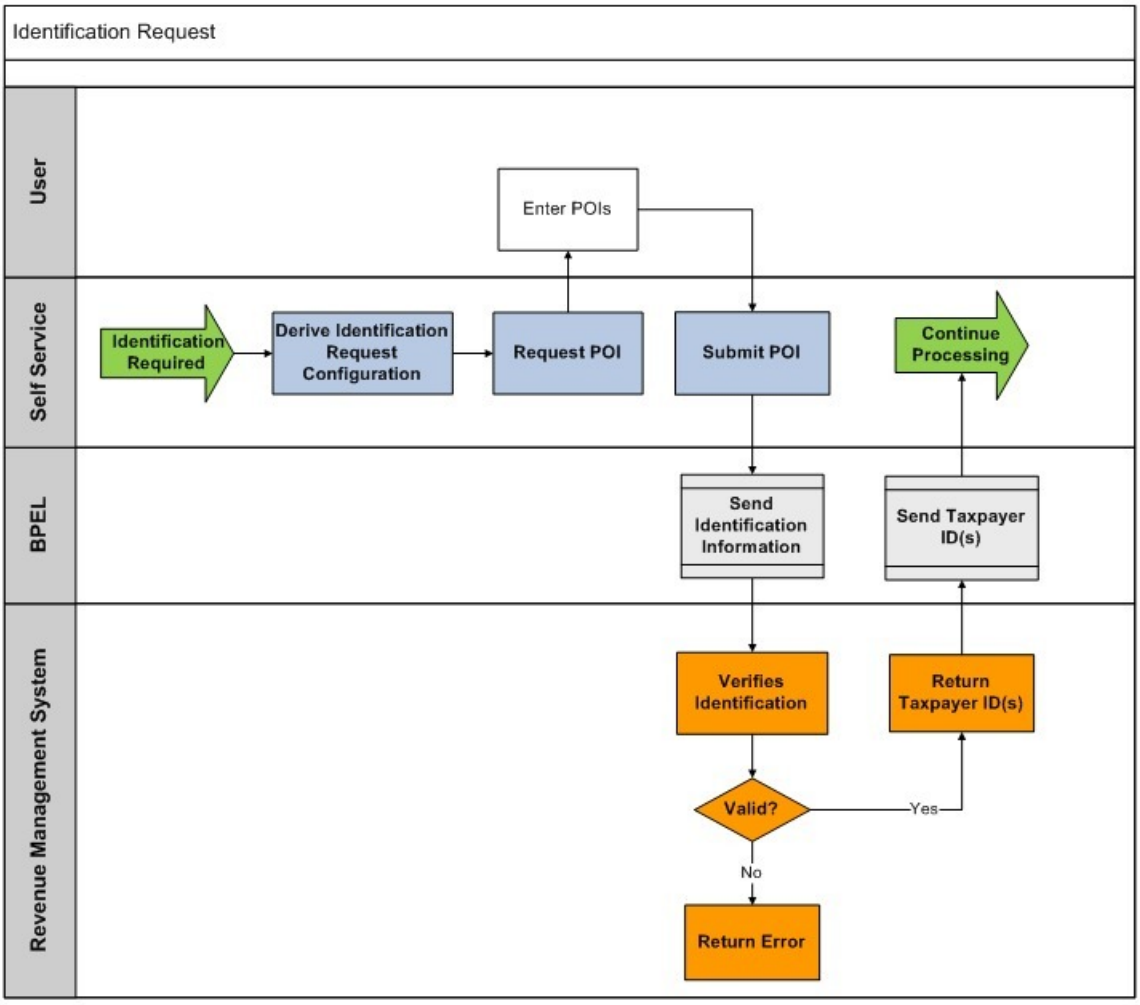
The subject and the sender's email address are derived from the email definition.

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.

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

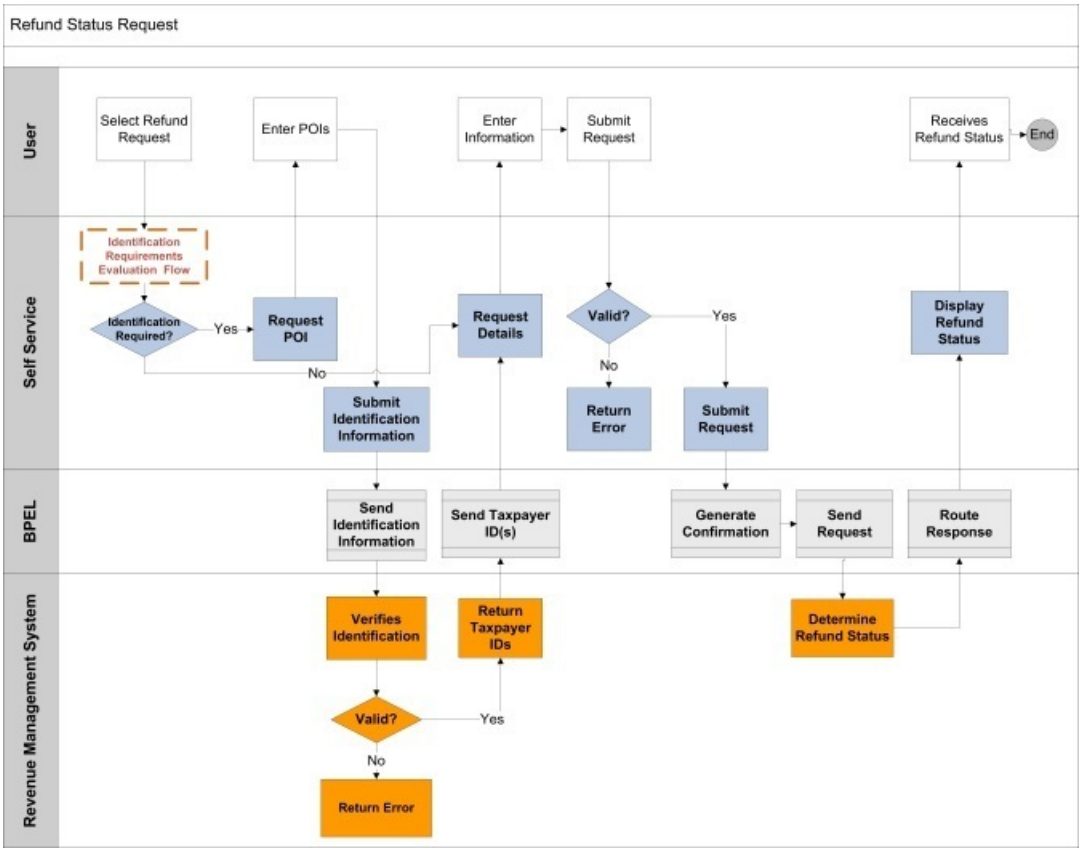
The taxpayer is asked to provide Proof of Identity (aka POI) details. The taxpayer enters the identification information and submits the request.

> **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 link. The taxpayer initiates the refund status inquiry by clicking on the associated link.

When the taxpayer clicks the link, 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.

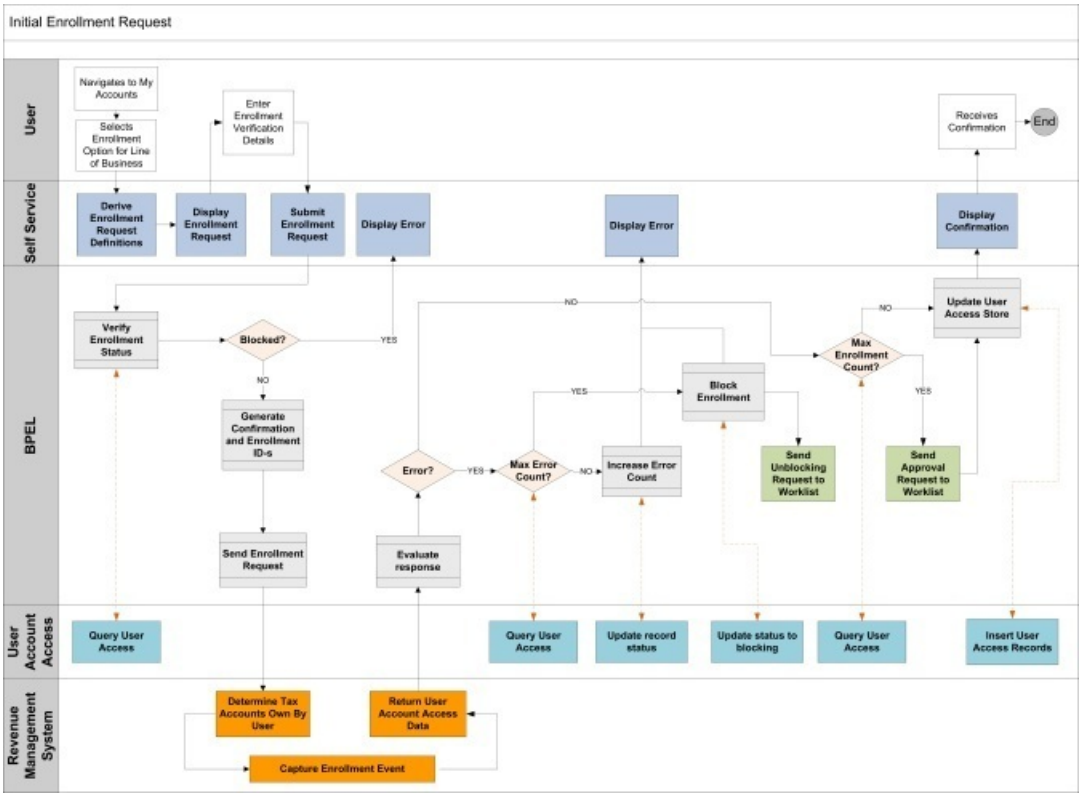
Confirmation

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.

User Enrollment Request

Process Flow

User Enrollment requests, which are invoked from the dedicated online page, belong to a special category of requests.



User Enrollment Request Selection

Enrollment is initiated from the **Enrollment Summary** page. On the initial screen, the description of each active service request of the special category **Enrollment Request** is displayed as a link. Enrollment request help text is displayed beneath the description. The taxpayer triggers the flow by clicking on the associated link.

When the taxpayer clicks the link, the system reads the Service Request configuration and renders the **Main Request Input** screen.

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

If the web service call is successful, the response contains the confirmation that is displayed to the user and the enrollment data is processed in the integration layer.

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 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.
- **Casual User Allowed?** Indicates whether a casual user may submit the request. For an Enrollment Request, this should be set to ***No*** .
- **Email Definition** is an optional reference to an email template. Not applicable for Enrollment Requests.
- **Line of Business** is required for Enrollment Service Requests. Only one Enrollment Request per Line of Business is allowed.

Processing

Category is used to group logically-related or similar requests. It can also represent a specific business feature, such as User Enrollment, Refund Status Inquiry or Taxpayer Identification. The categories are defined using the ***SRCAT*** lookup.

Response Mode controls whether the request will be sent synchronously to the Revenue Management System or placed into the queue (asynchronous mode). For Enrollment Requests, this should be set to **Synchronous**. For other requests, the 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. For an Enrollment Request, the method should be set to ***webservice.***

Recipient defines a support group dedicated to handling requests of this type. Not applicable for enrollment requests. For other types of requests, you can use the **RECIPIENT** lookup to maintain a list of support groups and their respective email addresses.

Identification Requirement

- **Identification Required** 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 unregistered (not signed in) taxpayers or should always be performed.
- **Identification Request** is a selection of taxpayer identification service requests. At run-time, the selected identification request is invoked and the 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 in which taxpayers 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:** Note: The e-mail 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 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 plain text.

Help appears on the **Service Central** screen page once the specific service request is selected. It can be entered as HTML or a plain text.

Help Override can be used by an implementation to replace the help text delivered with the product.

Service Request: Fields

This tab displays the Service Request's Fields list. Fields represent the information that should be collected in order to process this specific type of service request.

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

The ***Add*** button lets you define a new field.

Request Field

- **Request Sequence** uniquely identifies the field in the list. It is copied to the web service request fields collection and sent to the back-end system.
- **Field** references 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 internal use.
- **Request Help** provides alternative popup help text for the field on the screen.

Service Request Preview

The special ***Preview*** action is available on the Service Request maintenance page. Clicking this button lets you review the service request screen. You can fine-tune the screen design, the HTML of the header and footer; check field order, etc.

Service Request - Custom Properties

Custom Properties is a configurable collection of additional data elements that may be delivered to the integration layer and/or revenue management system. The collection can also be used to establish additional criteria for grouping Service Requests.

Property Type references a property type definition. Property Type controls the value type as either free-form or restricted by the specific Lookup value's list. It also defines whether or not the property is unique and searchable.

Property Value is the actual value of the property.

Copy To Service indicates if the property type and value should be copied to the web service XML.

Examples

Using Custom Properties for grouping:

- The agency would like to support multiple requests related to property taxes but want to group them by subject: tax bill questions, property ownership, tax liens, etc.

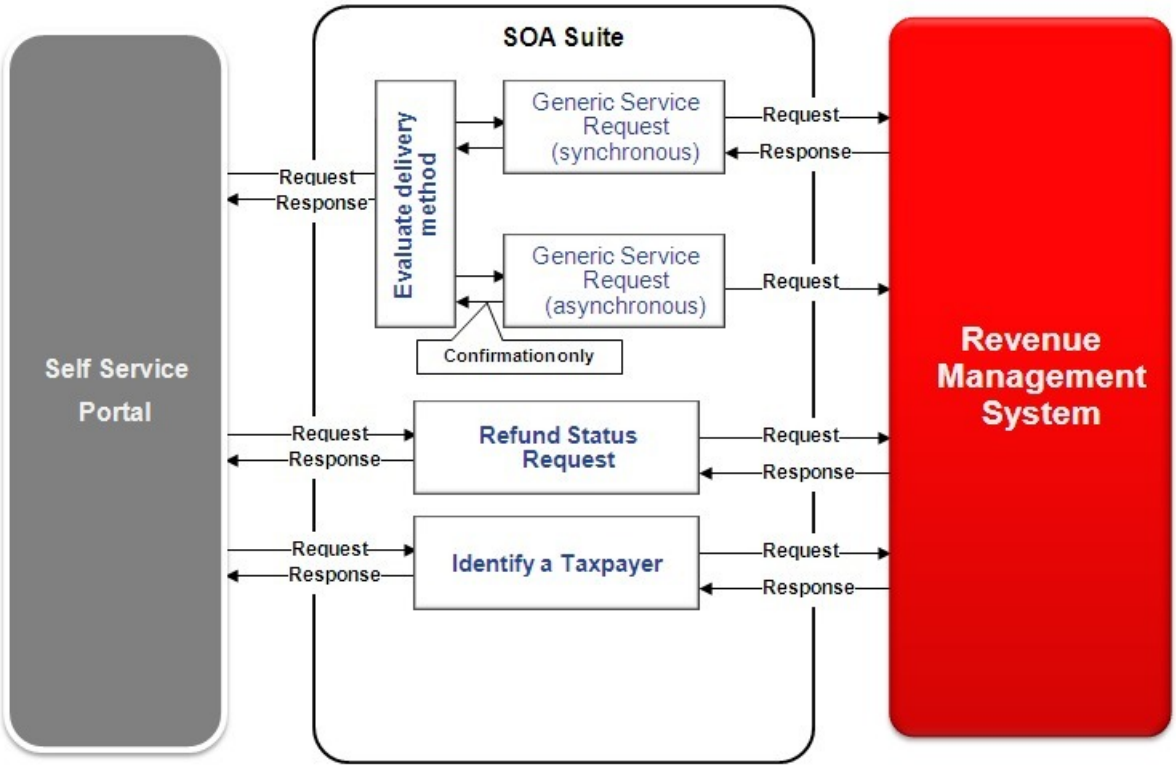
  Approach: Create a **Property Type** (value type Lookup, searchable, non-unique) and add a **Custom Property**(-ies) to service request(s) accordingly.

Using Custom Properties for message routing:

- A web portal consolidating all online self-services while taxes are maintained in multiple revenue management systems would require service requests to be routed into various destinations.

  Approach: Create a Property Type associated with a tax Type lookup, add a Custom Property to tax type-specific service requests, and mark the property as "Copy to the Service".

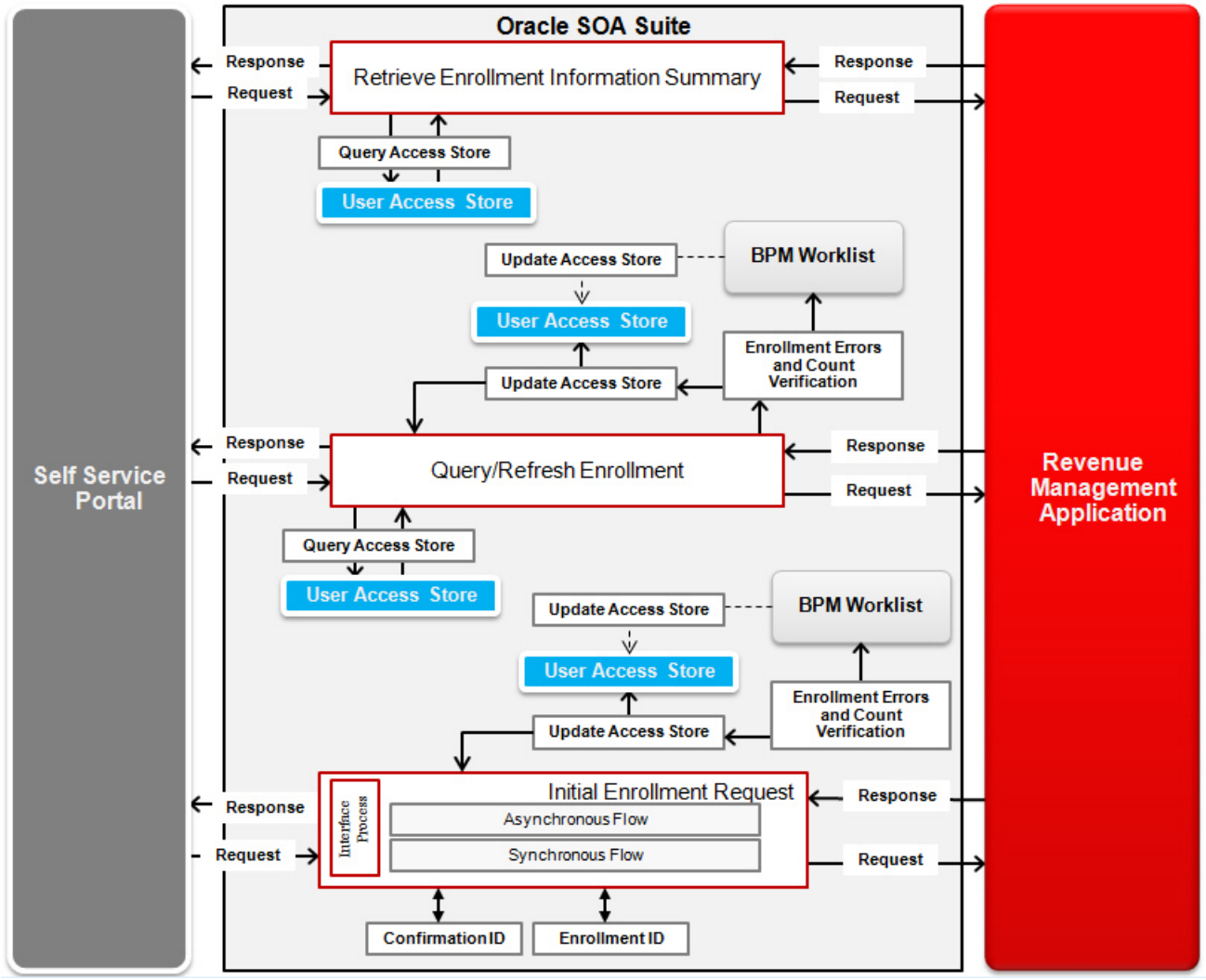
BPEL Processes



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.

Enrollment Request



The special features of the Enrollment Request integration flow are:

- A **Unique Enrollment ID** is generated by calling a utility using a dynamic partnerlink.
- The interaction with the **User Access Store** follows a successful response from the back-end system.
- Special provisions are enabled for fraud prevention.

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
TSTaxpayerServiceRequest	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. The request handling in the revenue management system could be as follows: • 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.
TSTaxpayerIdentification	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. If the taxpayer has been identified successfully, the response should be populated with one or more taxpayer IDs. The POI details evaluation may result in error. If so, the error message number should be populated on the response.
TSGetRefundStatus	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. The response expected to contain the following: • Confirmation message(s) describing the refund status. • Additional (optional) information: expected refund receive date and three amounts: refund, carry-over, and offset.

TSEnrollmentServiceRequest	This service is designed to evaluate input provided by the user and identify tax accounts owned by this user. The assumption is that initial enrollment results are recorded and the record(s) are traceable using the Enrollment ID provided on the request.
	The response is expected to contain the following:
	• Confirmation message(s) describing the enrollment status.
	• A list of enrollment entries; each entry includes:
	  • **Access Type**: code indicating the type of access "unit" (e.g., Account, Tax, Taxpayer). The value should have a corresponding definition in the Access Type table (Self-Service Portal Admin). DVM for the Access type code translation is provided.
	  • **Access keys**: name and value; up to 10 identifiers for the entity.
	  • **Access Role**: code identifying the level of access granted to the user. Should match one of the application roles existing in the portal application.
	  • **Enrollment status**. Valid values:
	  • **P – Pending**. User doesn't have online access to the accounts associated with the enrollment record in this status.
	  • **A – Approved**. User has online access to the accounts associated with the enrollment record in this status.

Supported Service Requests

Service Request	Description
TAX_CLEARANCE_CERT – Tax Clearance Certificate Request	Enable user to request and receive a tax clearance certificate. User is expected to enter the following information:

Service Request	Description
	• Email address (Alphanumeric) • Phone number (Alphanumeric) • Tax Certificate Purpose (Lookup) • Paper Copy Required (Boolean) Identification Requirements: IDENTIFY_TAX_CLEARANCE (Identification For Tax Clearance Request) Process synchronously, web service.
REFUND_STATUS_UNR – Check the status of your refund	Allow users to request a status of their refund for a given tax type and tax period. User is expected to enter the following information: • Taxpayer name (Alphanumeric) • Identifier Type (Lookup) • Identifier Number (Alphanumeric) • Tax Type (Lookup) • Filing Period Start Date (Date) • Filing Period End Date(Date) • Expected Refund (Number) Identification Requirements: none. Process synchronously, web service.
IDENTIFY_TAX_PAYMENT – Taxpayer Identification for Payment	Provides the ability to validate taxpayer POIs required to submit a payment and receive taxpayer's ID. User is expected to enter the following information: • 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) Identification Requirements: none. Process synchronously, web service.
IDENTIFY_TAX_ CLEARANCE - Taxpayer Identification for Tax Clearance	Provides the ability to validate taxpayer POIs required to request tax clearance. certificate and receive taxpayer's ID. User is expected to enter the following information: • Taxpayer name (Alphanumeric) • Date of Birth (Date) • Taxpayer Type (Lookup) • Identifier Type (Lookup) • Identifier Number (Alphanumeric) Identification Requirements: none. Process synchronously, web service.
TECHN-ISSUE – Report a website-related problem	Allows user to notify revenue authority about technical issues with the website.

Service Request	Description
	User is expected to enter the following information: • Taxpayer name (Alphanumeric) • Email Address (Alphanumeric) • Technical Issue – Topic (Lookup) • Detailed description of the problem (Text) Identification Requirements: none. E-mail.
ENROLL_BUSINESS – Access Your Business Accounts	Allows users to gain access to their business tax accounts. The user is expected to enter the following information: • Taxpayer ID Type (by selecting a value from the drop-down list based on the **IDTYPEBUS** Lookup). • ID Number (Alphanumeric). • Date of Birth (date). Identification Requirements: none. Line of Business: Business.
ENROLL_INDIV– Access Your Individual Accounts	Allows users to gain access to their individual tax accounts. The user is expected to enter the following information: • Taxpayer ID Type (by selecting a value from the drop-down list based on the **IDTYPEBUS** Lookup). • ID Number (Alphanumeric). • Date of Birth (date). Identification Requirements: none. Line of Business: Individual.

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
ENROLL_BUSINESS	10003, 30101, 30102, 30103
ENROLL_INDIVIDUAL	10003, 30101, 30102, 30103

Configuration

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

Service Request

Processing

Specify a recipient for each service request delivered via email.

Appearance

The service request description appears on the initial selection drop-down list. 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.
</span>
```

Help:

```
<span style="font-family: 'Arial'; "><b>Request assistance</b> finding something on the
 site, understanding our different file formats,<br/> printing files you've downloaded,
 installing or using the tax products CD-ROM, or any similar technical problem.</span>
```

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
- **IDTYPEIND** – Primary Identifier Type for Individuals
- **IDTYPEBUS** – Primary Identifier Type for Businesses

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

To 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 `"FIELD":"VALUE"` format, then entering 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 the self service portal application.

How To Enable a 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:

- The new active service request automatically appears 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**>.

How to Enable a New Enrollment Request

Design considerations

Create a new enrollment request if you have identified a category of website users who belong to a new line of business. Determine what information your revenue management system requires in order to identify the users' tax accounts, and grant the requested access. It is usually one or more data items, including dates and 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.

A new active enrollment request will automatically appear among the enrollment options. The request should be processed synchronously.

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

- The new active service request automatically appears as a hyperlink on the **My Accounts** 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 must be able to interpret this service request's set of input fields, identify tax accounts, and provide the response with access key sets, confirmation message(s) under **<confirmationData>**, 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 an 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

Enrollment

This section describes user enrollment feature implementation details.

For information on User Enrollment service request selection and configuration, see *User Enrollment Request*.

Overview

Enrollment vs. Registration

Self service portal registration is a process of establishing the secure login. As a result of the registration, a person obtains user name and password and able to log in into self service application and be properly authenticated.

Registration alone is not sufficient to gain an access to the tax account. The taxpayer should explicitly request access to his/her taxes and provides the information that may identify tax accounts owned by him/her in the revenue management system. The system evaluates the request and returns accessible tax accounts to the integration layer. Finally, the user/tax account association(s) are captured and become available for the account access verification purposes. This process is called enrollment.

A user enrolled into a tax account is able to view account financial information, file tax returns and make payments related to the account, update the taxpayer's contact and correspondence info, and request services.

Enrolled User

Getting Access to all Accounts

Single login provides access to multiple accounts managed by the Revenue Management Authority. Upon successful site registration, a user can log in and request access to the information and perform self-service operations on all accounts for

which the access was granted. In other words, an individual who also happens to be a business owner may log in once and view and manage both individual and corporate taxes.

Each line of business may require its own set of credentials as well as its own logic in identifying tax accounts owned by the user. The self service solution provides a flexible and configurable enrollment mechanism and allows configuration of a dedicated enrollment request per line of business.

Working with a Specific Account

After the enrollment process is completed successfully, a user may work within the context of a specific tax account, and the user's identification is no longer needed for payments, service requests or other online services. Tax account identifiers are populated on every outgoing web service request and delivered to both the integration layer and the back-end system.

Session Context

The entire time an enrolled user is browsing website contents and/or performing self-service operations, selected tax account identifiers are available in the session context. The user can switch accounts and select another one from the enrollment summary list.

The User Access Store

The user-account association is captured in a special table in the database. It is not exposed directly to the portal application and accessed only by SOA composites from the integration layer. During initial enrollment and subsequent enrollment refresh, the revenue management system creates an enrollment event and provides the account access information to the integration layer along with unique enrollment event ID. Integration layer is responsible for inserting new records into the access store.

The combination of user ID, enrollment event ID and Line of Business is unique; in other words, all enrollment records for the user for a specific line of business belong to the same enrollment event.

The **Access Type** and **Access Keys** combination should identify a logical data unit in the revenue management system, for example a tax account. Upon login, user account access information is retrieved from the store and user selects one unit at the time and works with this unit.

User account access data is stored as follows:

- Enrollment ID - unique identifier issued on the enrollment event.
- Line of Business
- Access Type
- Access Key 1 - Name
- Access Key 1 - Value
- .....
- Access Key 10 - Name
- Access Key 10 - Value
- Access Role - defines the user's access level.
- Status - defines the status of this particular access entry. (See valid values in the following table).
- Revenue Management System - code indicating the back-end application that maintains the actual account.

See the *PSRMSS Installation Guide* for additional details on the table definitions.

Enrollment Record Status

Enrollment records may be inserted and maintained in various statuses.

Status	Description
P	Pending Enrollment ID pending approval. User doesn't have online access to the accounts associated with the enrollment record in this status.
A	Approved User has online access to the accounts associated with the enrollment record in this status.
B	Blocked Too many subsequent errors; enrollment request has been denied.
H	Hold Indicates that the maximum allowed number of users has owner-level access to this record's account.
1....N	Enrollment In Error (intermediate statuses used internally) "Error count" status; used by the system to indicate and process subsequent enrollment requests after an error.

Initial Enrollment

A registered web site user that hasn't been enrolled yet starts the process from the Enrollment Summary ("My Accounts") page.

The page displays all available enrollment options, one option per line of business.

To obtain access to account(s), the user is asked to provide information that allows identifying accounts in the revenue management system.

The content of the enrollment request varies from one Line of Business to another. For each Line of Business, the implementation may configure a dedicated enrollment request.

An enrollment request is sent to the revenue management system to verify provided identification details. The system then returns access key sets, one set per tax account. Each key set contains at least one mandatory key uniquely identifying a taxpayer.

Access information is processed by the integration layer and stored in the user access repository, after which the user receives a confirmation message and can navigate to the Enrollment Summary page.

The user may always expand the enrollment scope by enrolling into accounts associated with an additional Line of Business.

Enrollment information is implicitly refreshed every time the user logs in and navigates to the Enrollment Summary page.

At this point, the self-service application requests enrollment data from the access store. The information is then forwarded to the back-end system so that the new access entries can be identified and retrieved. For example, if a user purchased a new property or filed additional taxes since the user's last login, the system may return new access information. That new information will be captured in the access store and shown on the summary.

Enrollment Event

To protect the user's enrollment information, a unique ID is generated in the integration layer and sent to the revenue management system with the initial enrollment request. Upon successful enrollment, each access "unit" in the user access store is stamped with the combination of Enrollment ID and Line of Business. The assumption is that the revenue management system captures and associates subsequent additions to the user's access within one line of business associated with the same enrollment event.

Enrollment ID remains a "shared secret" between user access store and the revenue management system and can be used for verification and reconciliation purposes.

Once the initial enrollment is successfully completed and the enrollment event is established, the system is aware of the fact that "user xxx is enrolled into his accounts for the line of business YYY with enrollment ID ZZZZZZZ". Having implicit enrollment refresh in place means that user won't need to request to enroll into same line of business again. In addition to simplifying enrollment process, this approach allows enrollment tracking and resolving enrollment issues.

Preventing Fraudulent Attempts to Enroll

Too Many Failed Attempts to Enroll

Enrollment is denied if the user submits several consecutive unsuccessful requests. This prevents automated "guessing" of POI information and protects the private and sensitive data stored in the revenue management system.

Too Many Users Requested Access to the Same Account

The system provides the ability to limit the number of users that may enroll into a single account. When the limit is reached, enrollment requests are held for the revenue management authority's review.

For additional information see *Enrollment Issues*.

Enrollment Summary

The Enrollment Summary shows all tax accounts accessible to the enrolled user.

Tax Accounts are grouped by Taxpayer. The taxpayer's name is returned by the web service.

The content of each entry is displayed according to the Access Type configuration.

The entry for each tax account displayed on Enrollment Summary includes three main components:

- **Summary Title -** the Summary Title Message text defined on the Access Type and the parameters returned by the web service.
- **Summary Details -** the Summary Details Message text defined on the Access Type and the parameters returned by the web service.
- **Location -** Formatted location information returned by the web service (optional).

The portal application expects one entry to be marked as a default tax account. This account is automatically loaded into the session context.

Navigation

The user can navigate to Taxpayer and/or Account Information pages from the Enrollment Summary page.

Enrollment Request

See *User Enrollment Request* for a description of the user enrollment request process flow.

Enrollment Options Selection

When an authenticated user navigates to the Enrollment Summary ("My Accounts") page, the *Get User Enrollment* web service is invoked. It retrieves the indicator whether the user is already enrolled or not. If the user is enrolled, Line of Business list is also returned. Based on these results, the page displays either of the following:

- **Non-enrolled user** sees a list of enrollment options - each entry contains an enrollment service request description and help;
- **Enrolled user** sees an Enrollment Summary, retrieved by invoking the *Get Enrollment Summary* web service and the list of Additional Enrollment Options (e.g., enrollment service requests associated with a Line of Business for which the user has not been enrolled).

Main Request Input

The main enrollment request screen is rendered dynamically, according the enrollment service request configuration. The taxpayer is prompted to enter the identification information required for enrollment within the line of business.

Upon request submission, a ***TSEnrollmentServiceRequest*** web service call is triggered. The request contains a collection of sequence/field name/field value entries.

If the web service call is successful, the response contains the list of access key sets and the confirmation.

Access information is processed in the integration layer and stored in the user access table.

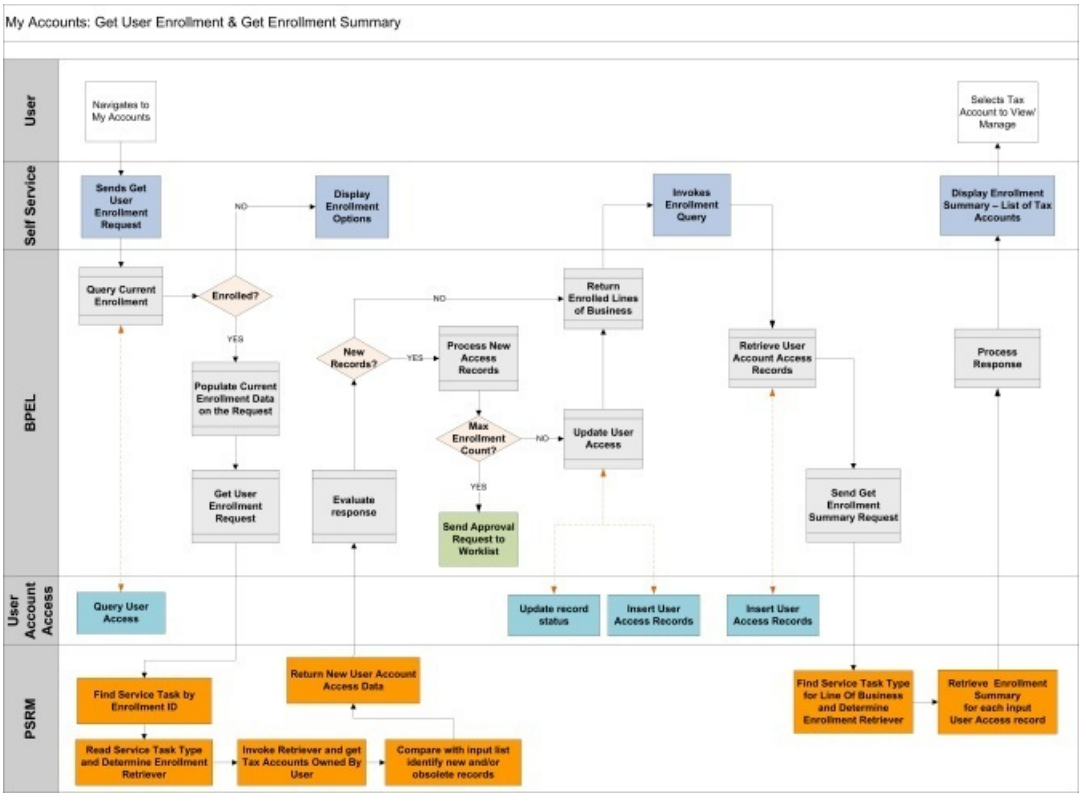
Confirmation information is returned to the portal application.

See the *Enrollment Issues* topic for more details on failed enrollment attempts handling.

Confirmation

The expected web service response contains a confirmation number and message(s). This information is displayed to the user. The user is then redirected back to the Enrollment Summary page.

Enrollment Refresh and Summary



User access information is implicitly refreshed upon login. The Get User Enrollment web service request is initiated on load when the user navigates to the Enrollment Summary page.

- The SOA Composite is querying the user access store, retrieves all enrollment records for the input user ID and populates them on the request; each record includes tax account identifiers, line of business and enrollment ID.
- The request is forwarded to the revenue management system that is expected to evaluate the list, identify new tax accounts owned by the user and/or tax accounts that should no longer be associated with the user.
- The response is processed by the integration layer that updates the user access store.

The results of a successful enrollment refresh may be a creation of one or more new records in user access store. Each record is stamped with the original enrollment ID/line of business combination, meaning it belongs to the original enrollment event.

Enrollment Summary

After an enrollment request is completed, the user is redirected to the Enrollment Summary page. This immediately performs an enrollment refresh and then invokes the *Get Enrollment Summary* web service.

The *Get Enrollment Summary* web service retrieves the Approved access key(s) from the User Access Store and passes them to the revenue management system, which in turn derives information to be displayed on the page.

Formatted Enrollment Summary information for each tax account is displayed to the user.

At this point, the tax account (access keys set) that is marked as the default on the list is automatically loaded into session context.

The selected account and taxpayer info is always shown on the top right corner of the main page area.

Enrollment Issues

The following topics describe enrollment issues that may require preventive implementation measures.

Too Many Failed Attempts to Enroll

Too many failed enrollment requests coming from the self service may suggest a fraudulent attempt to "guess" taxpayer identification details and gain access to the taxpayer's account. The system provides the ability to block excessive numbers of subsequent failed attempts. It is assumed that only one enrollment event per line of business exists. The diagrams below illustrate the flow of events:

- After the first failed attempt to enroll, the system creates an enrollment record in error and captures it in user access store. The record is stumped with enrollment ID, user ID and the line of business.
- Each subsequent failure attempt is counted and the status of the record is updated.
- When the count reached the maximum allowed, the enrollment record becomes blocked.
- The unblocking request is sent to the Worklist application. The tax authority security administrator may now login and review the request, contact the taxpayer and take other necessary actions. The unblocking request may be approved or rejected.
- If the enrollment request is unblocked, the user may repeat an enrollment attempt.

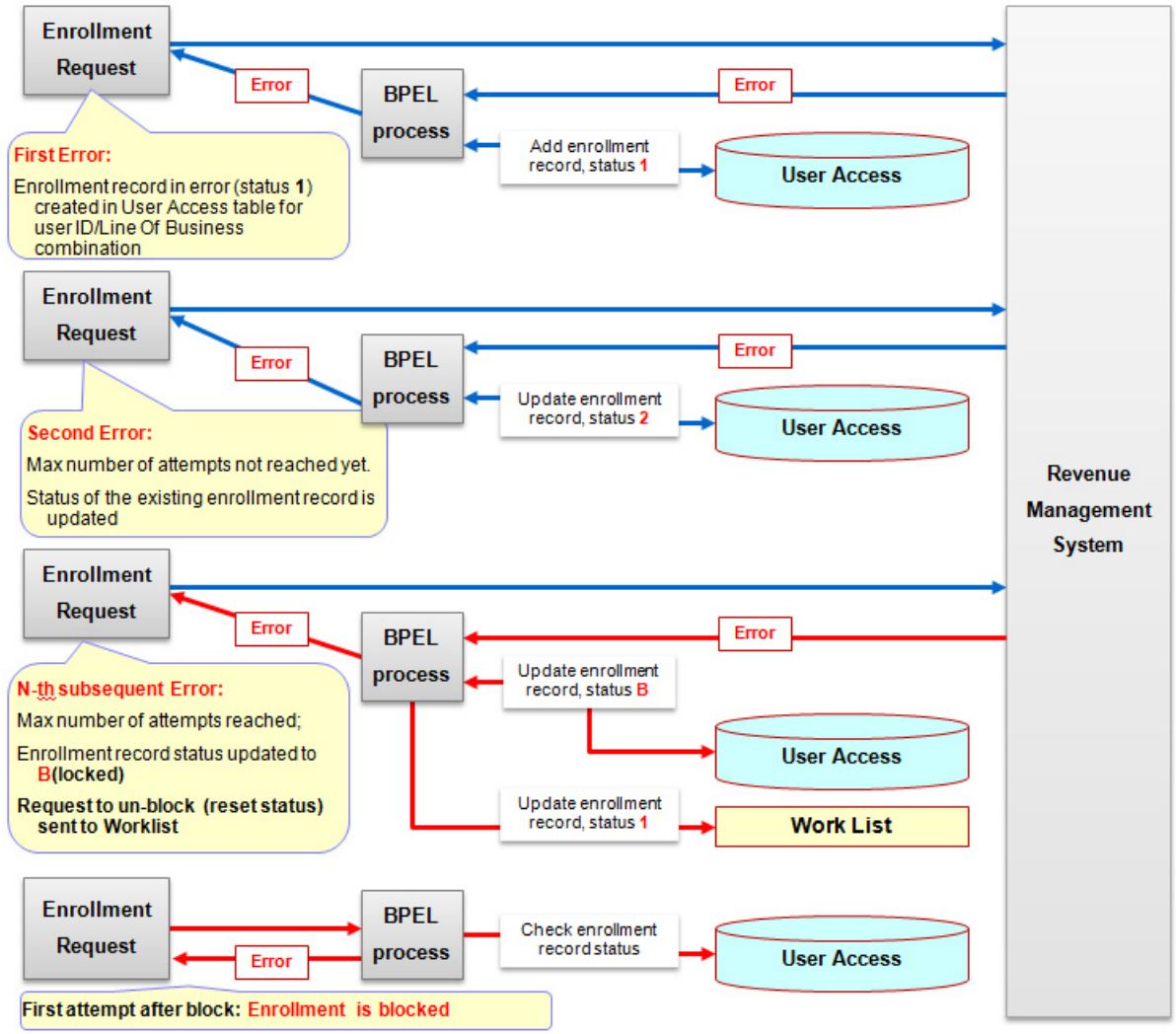


Figure 3: Blocked enrollment after excessive failed enrollment attempts

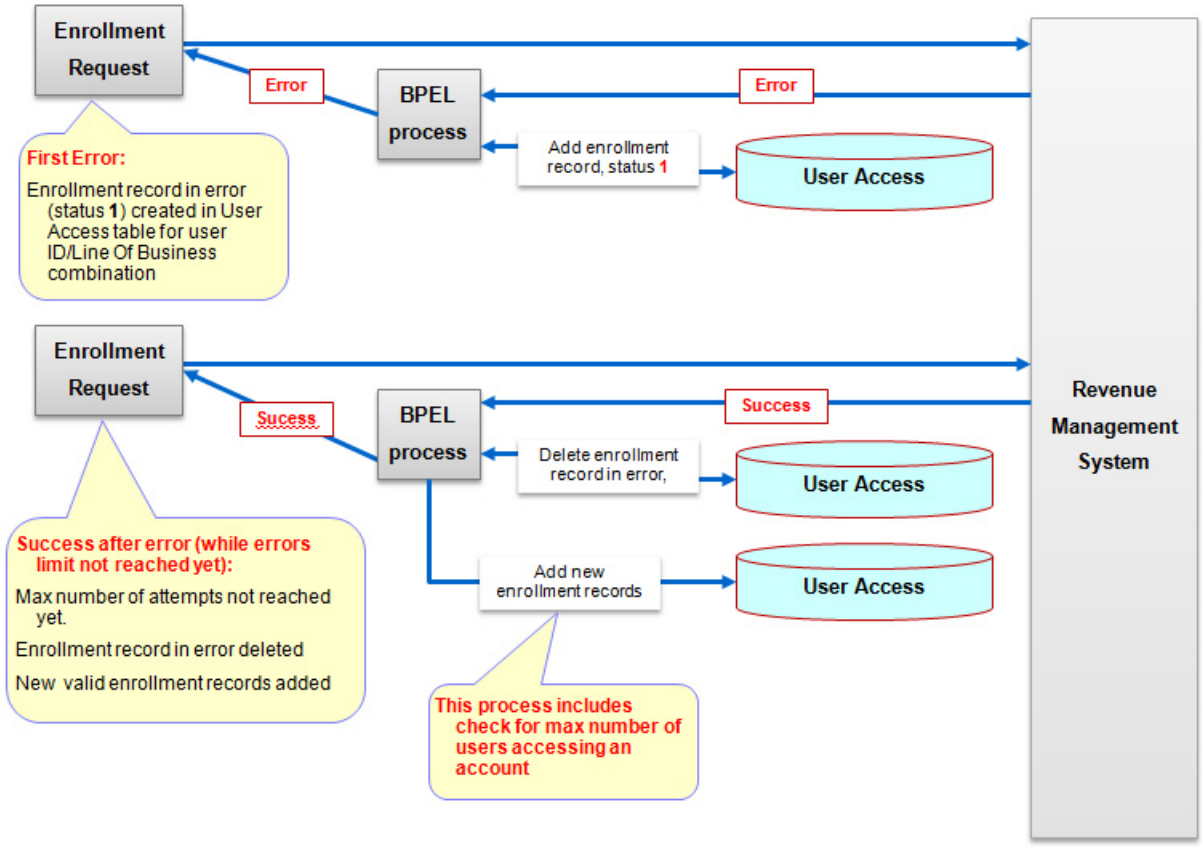


Figure 4: Handling successful access after failed enrollment attempt

See *Prevent Fraudulent Enrollment* for additional information on this issue.

Too Many User Account Access Requests

Too many requests to access the same tax account may suggest of the attempt to create the situation where it is possible to overwhelm the system and trigger performance problems. Consider the scenario where thousands of users are trying to access the same data simultaneously.

Even if the request to access the account is coming from a legitimate owner, the tax authority may wish to restrict the number of owners accessing the information. The product supports an ability to place enrollment request on hold and allow manual review of a specific taxpayer's situation.

The diagram below illustrate the flow of events:

- If the enrollment is successful, each tax account identifiers set is evaluated. The system counts access to the same tax account belonging to other enrollment event. If the count is higher than the maximum allowed, the entry is added to the user access store in status On Hold and the approval request is forwarded to the Worklist application.
- The tax authority security administrator may now login and review the request, contact the taxpayer and take other necessary actions. The unblocking request may be approved or rejected.
- If the enrollment request is approved, the entry status is changed to Approved, the associated tax account appears on the enrollment summary.

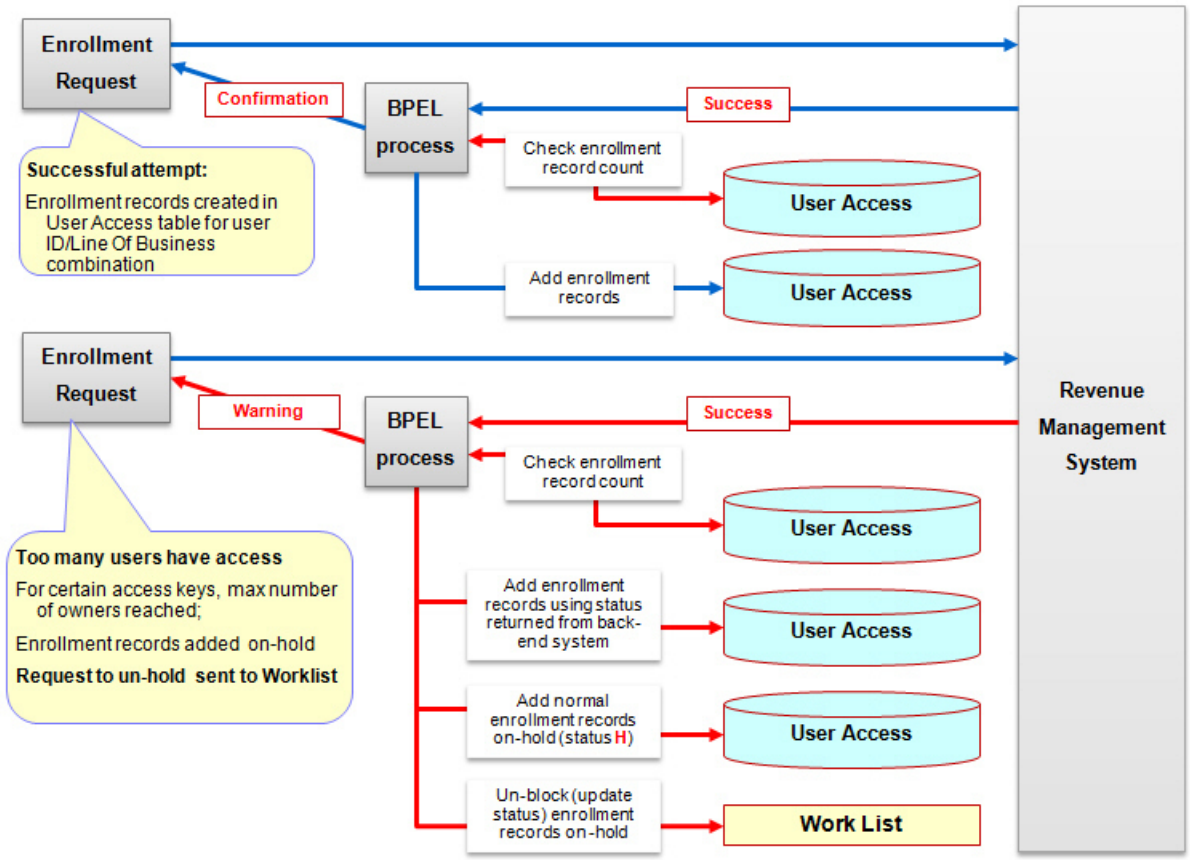


Figure 5: Blocking enrollment after excessive user account requests

Review Enrollment Issues

As a result of the enrollment issues multiple messages may end up in the Worklist queue. There are two types of requests that require human intervention and manual review:

- A request to change the status of user access record from Blocked (B) to First Error (1). A request of this type contains enrollment ID, user ID and the line of business. The blocking record is a result of a failed enrollment request and does not include tax account identifiers.
- A request to change the status of user access record form On Hold (H) to Approved (A). The request of this type contains enrollment id, user id, line of business, tax account identifiers (access keys and access type) and revenue management system code.

Essentially, the task in the Worklist is a web service request "stopped" on its way from one node to another; in this case, from the SOA Composite to the utility that updates the user access store table.

The reviewer uses the Worklist application to pick a task from the list, view request contents, and either approve or reject the task. Approval means the request completes the interrupted flow; rejection means the request is cancelled.

> **Note:** To make an informed decision about approval, the enrollment administrator may need access to sensitive customer data stored in the revenue management system. Consider setting up a dedicated user group/security role for agency users responsible for enrollment issues.

See *Oracle BPM Worklist* documentation for more details.

Defining Enrollment

Line of business and access type are referenced in multiple tax account-related configurations and control the appearance and contents of the enrollment summary. The enrollment service request configuration defines the input that the taxpayer should supply in order to obtain access to tax accounts and define how the input is presented in the user interface.

Line of Business

Line of Business is defined using the customizable Lookup ***LINEOFBUSINESS*** .

Select **Settings > Lookup** to maintain the LINEOFBUSINESS lookup.

The base product is delivered with two values: **Individuals** and **Business**.

Access Type

See *Access Type* topic in the Configuration section of this document for details on configuring the Access Type for User Enrollment.

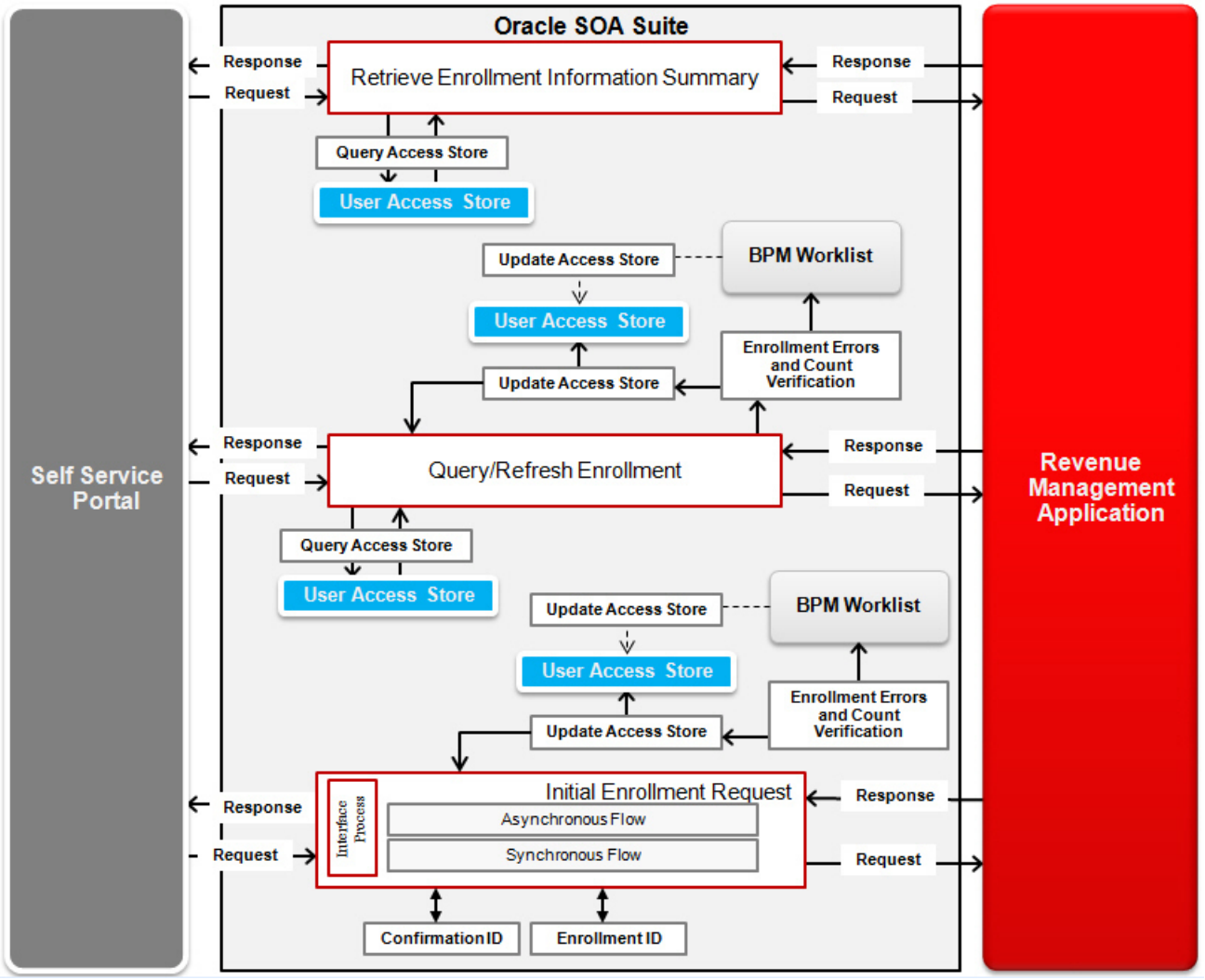
Service Request

Select **Configurations** > **Service Request** to configure an enrollment service request.

See the section *Defining Service Requests* for complete details.

BPEL Processes

Integration Overview



BPEL integration process flow diagram

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 and enrollment ID are generated when an enrollment request is invoked. The confirmation number is returned as part of the response. The Enrollment ID, which is used for various aspects of enrollment verification, is captured in the user access store and is also assumed to be captured in the back-end system as a unique identifier of the enrollment event.

Preventing Fraudulent Enrollment

The SOA composite that processes the **TSEnrollmentServiceRequest** web service performs the following special logic:

- Prevents multiple failed enrollment attempts. The maximum number of subsequent failed attempts allowed is defined in BPEL `configuration.properties`.

- Places the enrollment for a specific tax account(s) available for tax authority review. The max number of users allowed to access an account is defined in BPEL `configuration.properties`.

Implementing Enrollment

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

Web Services

To support user enrollment, the revenue management system is expected to implement the web services described in the following table.

Web Service	Description
TSEnrollmentServiceRequest	Based on the information collected from the taxpayer on the web portal, this service returns a list of access key sets. Each entry contains information sufficient to identify a tax account in the system. It should also return a confirmation or error with the response. The request message contains line of business, a service request type (code), and a collection of sequence/field name/field value details. User access data is processed in the integration layer and stored in the user access store. Request handling in the revenue management system is designed to work as follows: Determine the service request type. Determine tax accounts the user should be able to access using input line of business and request details field values. Populate the response with access key sets. Each set contains: Enrollment ID Access type that describes the purpose of this access key set; for example, ACCOUNT or TAX ROLE Access keys (name and value) Role - should be a value corresponding to one of the security user roles defined in the portal application Confirmation message(s) numbers and parameters
TSGetUserEnrollment	This service evaluates current user enrollment information, performs implicit refresh of user enrollment data, and returns the enrollment overview to the portal application. The new access keys form the response and are processed in the integration layer. The list of Line of Businesses for which the user is enrolled are returned to the portal. Request handling in the revenue management system is designed to work as follows: Find enrollment event(s) associated with input enrollment ID(s); then check if the user is permitted access to additional new tax accounts. Return new access key sets.
TSGetEnrollmentSummary	This service is responsible for retrieving user-friendly information to be displayed on the Enrollment Summary page The request contains a list of access key sets. For each access "unit", the response is expected to contain: Taxpayer name and account name Summary title and summary details; it could be either a finalized and formatted string (text/HTML) or a list of parameters that can be injected into a message defined in the portal app for each Access Type

Web Service	Description
	An indicator for default access entry, marking an access "unit" to be selected on page load

Supported Enrollment Requests

Service Request	Description
ENROLL_INDIV - Individual Enrollment Request	Enables the taxpayer to request and receive a tax clearance certificate. The taxpayer is expected to provide the following information: • Select an Identifier Type from the drop-down (based on lookup IDTYPEIND) • Identifier (Alphanumeric) • Date of Birth - date Identification Requirements: none Process synchronously, web service
ENROLL_BUSINESS - Business Enrollment Request	Enables the taxpayer to enroll and manage business tax accounts. The taxpayer is expected to provide the following information: • Select an Identifier Type from the dropdown (based on lookup IDTYPEBUS) • Identifier (Alphanumeric) Identification Requirements: none Process synchronously, web service

Supported Access Types

Access Type	Description
TAXROLE - Tax Account	Access entity is identified by two keys: PER_ID - taxpayer ID TAX_ROLE_ID - tax account ID **Line of Business - Business (BUS)** Enrollment Summary Title Message 30000 Enrollment Summary Details Message 30002 Account Summary Title Message 41101 Account Summary Details Message 41100 **Line of Business - Business (IND)** Enrollment Summary Title Message 30000 Enrollment Summary Details Message 30001 Account Summary Title Message 41101 Account Summary Details Message 41100 For this configuration, the revenue management system is expected to return the following information as enrollment summary parameters: For Title: Tax Type; value corresponding to one of the values of TAXTYPE *Lookup* For Details: Tax Account start and end dates (if available) Outstanding Balance

Access Type	Description
	"As of" date for the balance calculation
	Location (address) - for Business tax types

Messages

The following messages are defined to support enrollment functionality:

Component	Messages
Enrollment Summary	30000, 30001, 30002, 41100, 41101

Configuration

To enable the supported service requests you may add/modify the following configurations.

Lookup

Add values for Lookups referenced by enrollment request fields. Examples:

- ***TAXPAYERTYPE*** - Taxpayer Type. For this lookup, add the values corresponding to those in the revenue management system.
- ***IDTYPEBUS*** - Identifier Type for Business.
- ***IDTYPEIND*** - Identifier Type for Individuals.
- ***STATE*** - State.
- ***TAXTYPE*** - The Tax Type. In order to display tax account information properly, populate this lookup with values used in the revenue management system, providing appropriate descriptions for each.

Additional lookup values are available.

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

Example:

One of the Enrollment Summary Title parameters is Tax Type. This parameter references a Lookup TAXTYPE. No values are provided with the base product. Implementers should decide which tax types should be available for the taxpayer, and add these tax types as new lookup values.

Messages

Several info/error messages are provided for enrollment requests (see *Supported Service Requests* and *Enrollment Summary*). Add the DVM mapping for the messages you expect to receive from the revenue management system. You can also define new messages and use them instead of those provided with the base product.

BPEL DVM Mapping

OTSS_ServiceRequestType

An entry is provided for each enrollment request included in the base product. The column ***OTSS_SRType*** contains enrollment 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 enrollment request code, leave the translation value blank. As a result, the **<serviceRequestType>** node on the request XML will be empty.

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

Examples:

The content of the node:

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

With a translation value of "XXX" for ENROLL_BUSINESS:

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

With a blank translation value for ENROLL_BUSINESS:

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

Without an entry in the DVM for ENROLL_BUSINESS:

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

OTSS_FieldCodes

This DVM translates 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. (Note: This translation is optional.)

Examples:

The content of the entry on **<requestField>** list for the request field **ID_TYPE** with value ***IND*** selected from the lookup **IDTYPEIND:**

```
<requestField>
<sequence>
<fieldName>ID_TYPE</fieldname>
<fieldValue>SSN</fieldValue>
</requestField>
```

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

```
<requestField>
<sequence>
<fieldName>ID_TYPE</fieldname>
<fieldValue>XXX</fieldValue>
</requestField>
```

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

```
<requestField>
<sequence>
<fieldName>ID_TYPE</fieldname>
<fieldValue></fieldValue>
</requestField>
```

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

```
<requestField>
<sequence>
<fieldName>ID_TYPE</fieldname>
<fieldValue>SSN</fieldValue>
</requestField>
```

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 Enrollment Request

> **Design considerations**

- Create new enrollment request if you identified a category of website users who belong to a new line of business. Determine what information your revenue management system requires in order to identify user's tax accounts and grant the requested access. It is usually one or more data items, including dates and amounts. You can also offer a selection of predefined 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's optional. Design the additional content (help, header and footer) for the user interface.
- You may decide that a stand-alone identification step is not required and incorporate taxpayer identification details into the service request fields collection.
- A new active enrollment request will automatically appear among the enrollment options.

Configuration

- Configure the enrollment request according to the design.
- In the SOA/BPEL layer, add the mapping for the service request type code and specific fields (if applicable). Also add the mapping for the response message.

Implementation

- The revenue management system must be able to interpret this service request's set of input fields, identify tax accounts, and provide the response with access key sets, confirmation message(s) under **<confirmationData>,** or an error message under **<errorMessage>**.

Chapter 6

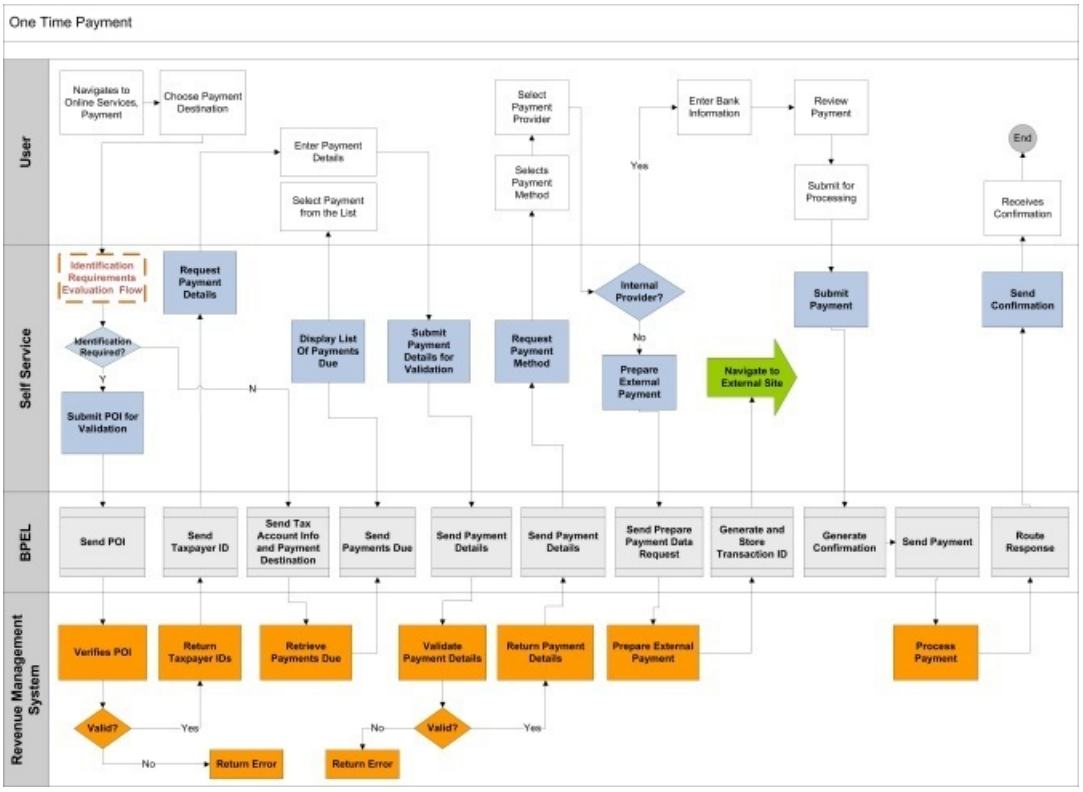
One-Time Payments

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. The ability to make a certain type of payment may be offered to all website users or exclusively to registered and signed-in users. This restriction is controlled by payment destination configuration.

Identification Requirements Evaluation

The system verifies that the user has already been authenticated (logged in) and checks whether a specific tax accounts has already been selected. If no account has been selected, the user is prompted to navigate to the My Account page and make a selection; with no access keys in context, the user is now treated as a casual user.

The system reads the Payment Destination configuration. It determines whether this type of payment requires taxpayer identification. The possible scenarios include:

User Registered	Enrolled/Account Selected	Identification Required	Result
Yes	No	Yes	Identification invoked
Yes	Yes	Yes	Payment Flow invoked
Yes	No	No	Payment Flow invoked
No	N/A	Yes	Payment Flow invoked
No	N/A	No	Payment Flow invoked

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.

Retrieving Payments

The one-time payment process includes this step under the following circumstances:

- The user is logged in.
- The user navigated to the **My Accounts** page and selected an account to work with.

The list of outstanding payments due for the selected tax account is retrieved from the revenue management system and presented to the user. For example, if an enrolled user chooses to pay an amount due in a filing period, the system will retrieve all filing periods for which payments are due, along with payment amounts.

At this time, a web service, **TSRetrievePaymentsDue**, is triggered and the request containing access keys and payment destination is sent to the revenue management system. The successful response contains a collection of payments due.

Each entry on the collection contains payment amount and a list of payment summary parameters. These parameters are substituted on the message defined on payment Destination and the formatted payment info is displayed on-screen.

Instead of entering payment information manually, the user may select a payment from the list and navigate to the payment details page with all data pre-populated.

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 **TSOneTimePayment** 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 TSProcessExtPayReportRecord 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) website users.

Line of Business (optional) allows associating payment destination with a specific taxpayer category.

Category provides a way to define a special use for the payment destination. Those in the General category appear on the Make a Payment page; those in the Form category are used for payments performed as part of the online form filing process.

Casual User Allowed indicates if this payment destination applies to casual users. If unchecked, the destination won't appear on the payment options list if the user is not signed in.

Default Payment Amount provides a way to set a fixed charge for the payment option. The amount will be the default on the Payment Details page and may be manually overridden at run-time.

Appearance

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

Summary Message is a message whose text is used to display formatted information on the Payments Due list. Summary parameters are retrieved by the **Retrieve Payments Due** web service.

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**.

Payment Destination - Custom Properties

Custom Properties is a configurable collection of additional data elements that may be delivered to the integration layer and/or revenue management system. It can also be used to establish additional criteria for grouping Payment Destinations.

Property Type references property type definition. Property Type controls value type as either free-form or restricted by the specific Lookup value's list. Property Type also defines whether the property is unique and searchable.

Property Value is the actual value of the property.

Copy To Service indicates if property type and value should be copied to web service XML.

Examples

Using Custom Properties for grouping:

- In a multi-jurisdiction agency, certain payment options are specific to a township. Approach: Create a lookup whose values represent towns and townships, create a **Property Type Township** (value type Lookup, searchable, non-unique) and add a **Custom Propert**y(-ies) to payment destination(s) accordingly.

Using Custom Properties for message routing:

- A web portal consolidating all online self-services while taxes are maintained in multiple revenue management systems would require payments to be routed into various destinations. Approach: Create a **Property Type** associated with a tax Type lookup, add a **Custom Property** to tax type-specific payment destinations, and mark the property as "Copy to the Service".

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**.

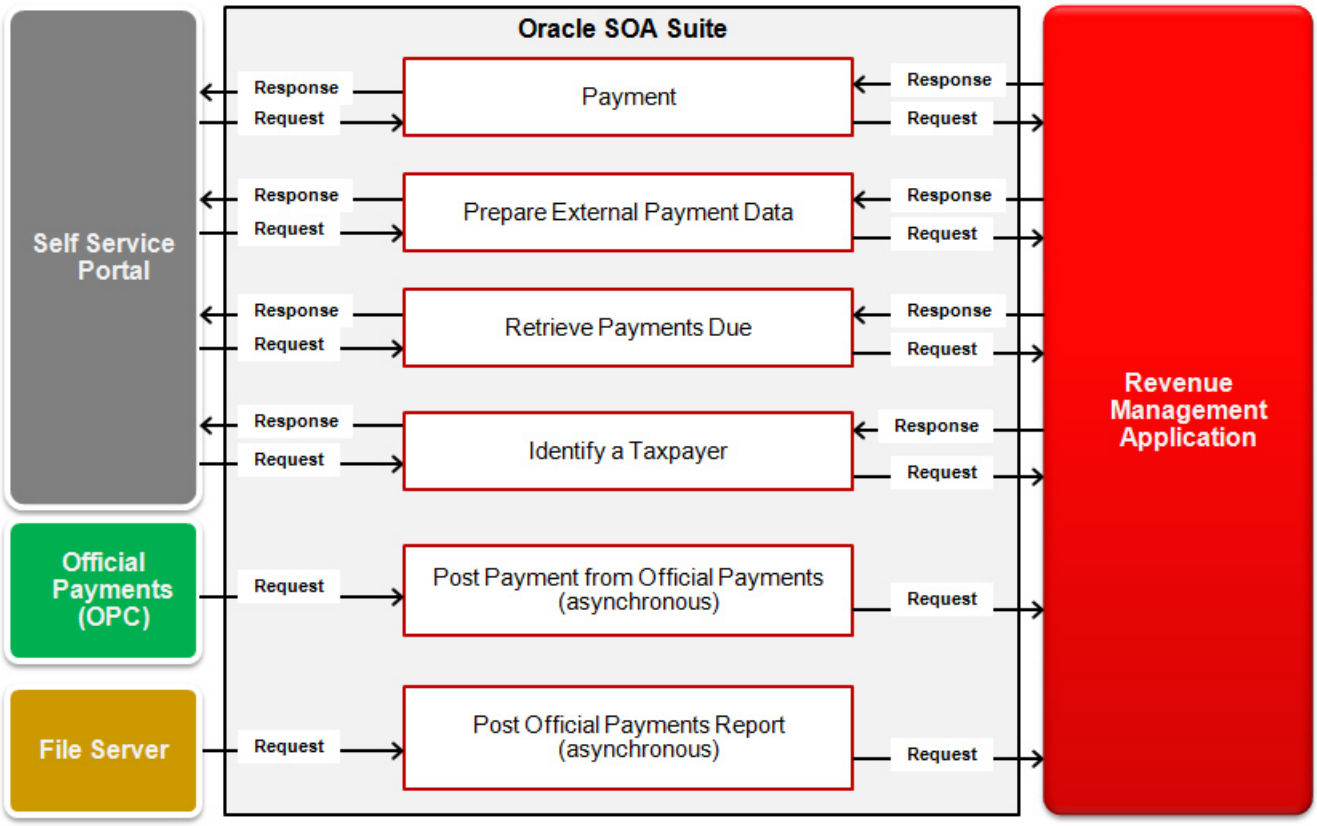
On the Payment Selection Page the taxpayer selects a payment method from the dropdown, and all active providers that support this method are displayed.

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.

Refer to the *BPEL/SOA Integration* chapter for detailed information about integration layer implementation

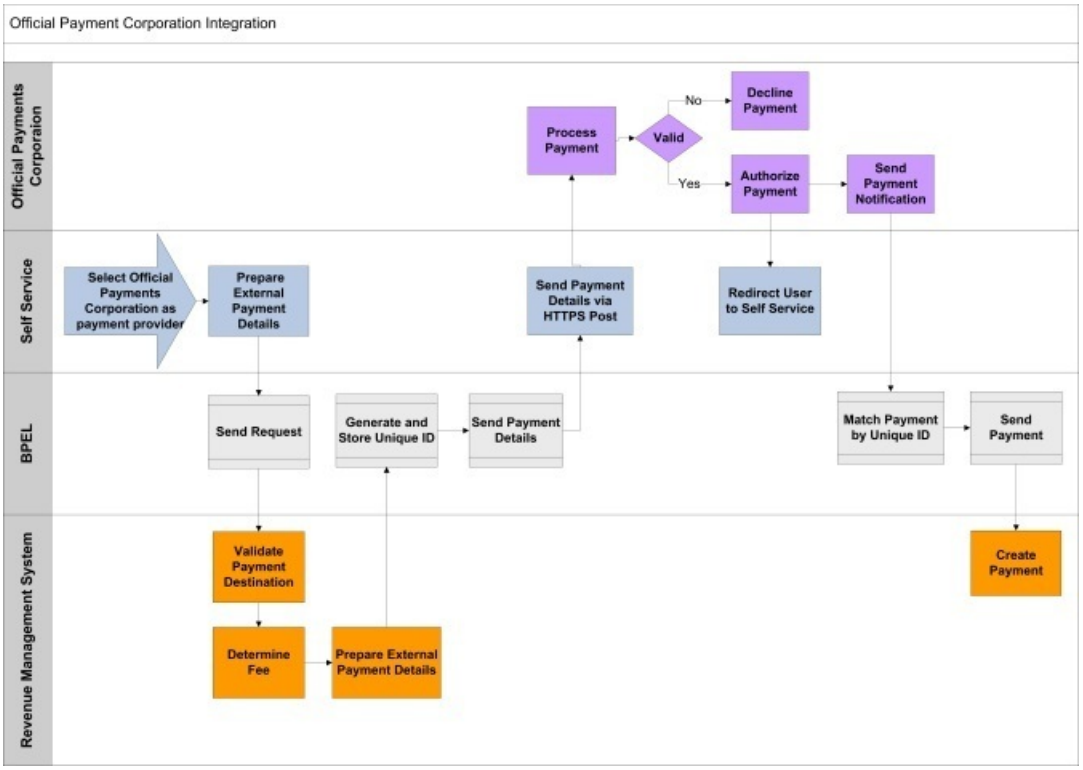
Integration with Official Payments Corp.

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 c**de-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 filters 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 **TSOneTimePayment** 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.

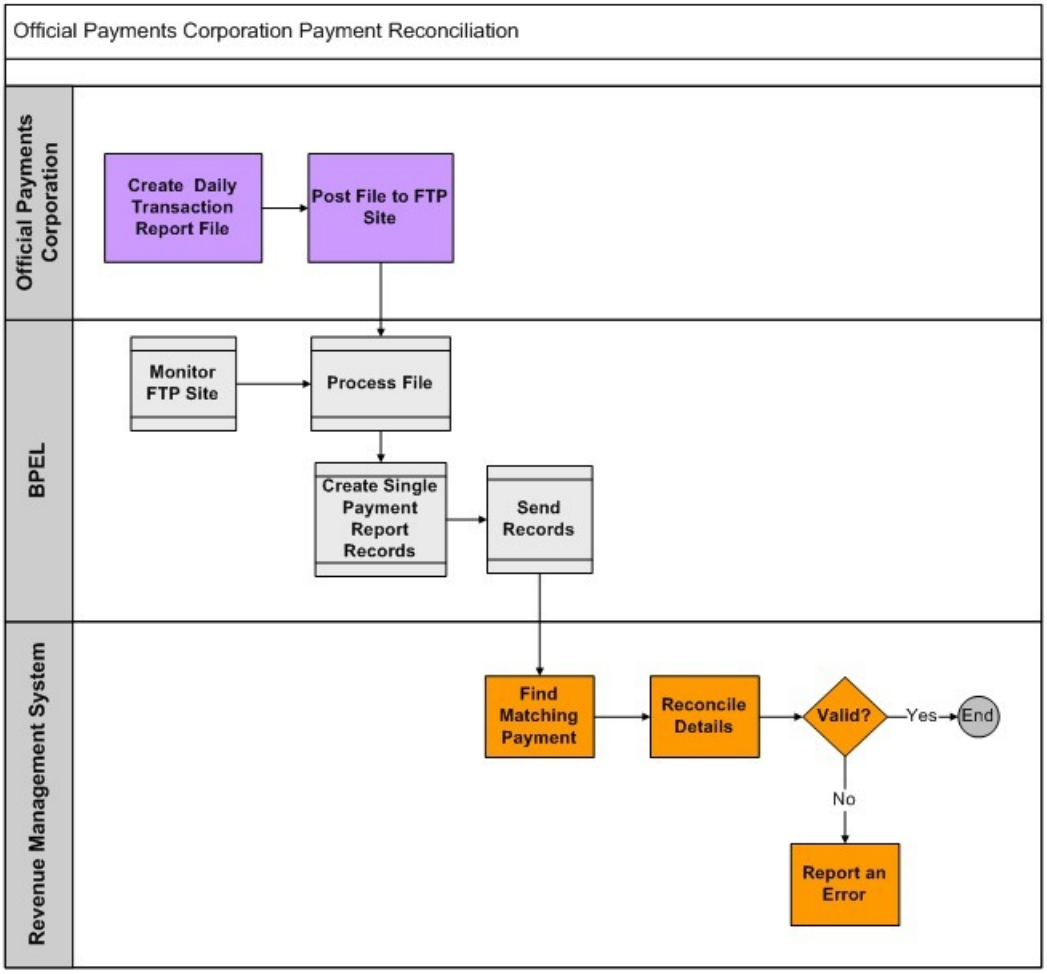
Implementation may use the following methods to advise the taxpayer about this option:

- 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 **TTSProcessExtPayReportRecord**.

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 **TTSProcessExtPayReportRecord** 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 the Official Payments testing site https://staging.officialpayments.com/pc_entry_cobrand.jsp as the Re-direct URL. Note that this is a temporary definition and should be changed later to point to the 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 48193674838312020873264734205247 3017 for payments whose convenience fee is paid by the taxpayer.

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

- NFEE - 27362262501965139940530820135778488
- CNVF - 48193674838312020873264734205247 3017

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 **OTTSS_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 **TSPrepareExtPaymentData**.

Example:

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

Assuming that the **<convenienceFee>** node on **TSPrepareExtPaymentData** 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
TSOneTimePayment	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:

Web Service	Description
	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.
TSProcessExtPayReportRecord	The request includes the single payment report information. It is expected to trigger payment reconciliation logic in the revenue management system. No response is expected.
TSRetrievePaymentsDue	The service is invoked only when user is signed in and enrolled. The request contains user information, access type and access keys and payment destination. The response is expected to contain collections of payments due retrieved using the input. The logic is expected to identify all payments of the input "type" due as of the current date for the access "unit" identified by access type and keys. For example, for payment destination "Tax Obligation" and Tax Account 12345678, it should identify all unpaid tax obligations. Each entry on the collection contains amount, currency, and a list of summary parameters. The parameters would be injected into summary message defined on payment destination in order to display formatted payment due info on the screen.

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 to 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 Extention 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 woudl 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 mehtod 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).

Payment Destination	Description
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:

```
11001, 11002, 11003, 11004, 11005, 11006, 11007, 11008, 11009, 11010, 11101
```

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.
- You can alter the description of the payment on the **Payments Due** list by changing the summary message text (using the message text override).

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.

Custom Properties

Use custom properties to communicate additional data/configurations to the integration and revenue management layers.

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 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 7

Taxpayer Information

Overview

The Taxpayer Information page is accessible to registered & enrolled self-service users. It consolidates the most important information about the taxpayer: *Taxpayer Summary*, *Contact Information* and *Correspondence Information*. From this page, the user may edit phone number(s), email address and a collection of various addresses.

All the data is retrieved for the taxpayer currently in context. The ***View Account*** link allows direct navigation to the **Account Info** page. The user should navigate to the Enrollment Summary page in order to switch to a different taxpayer.

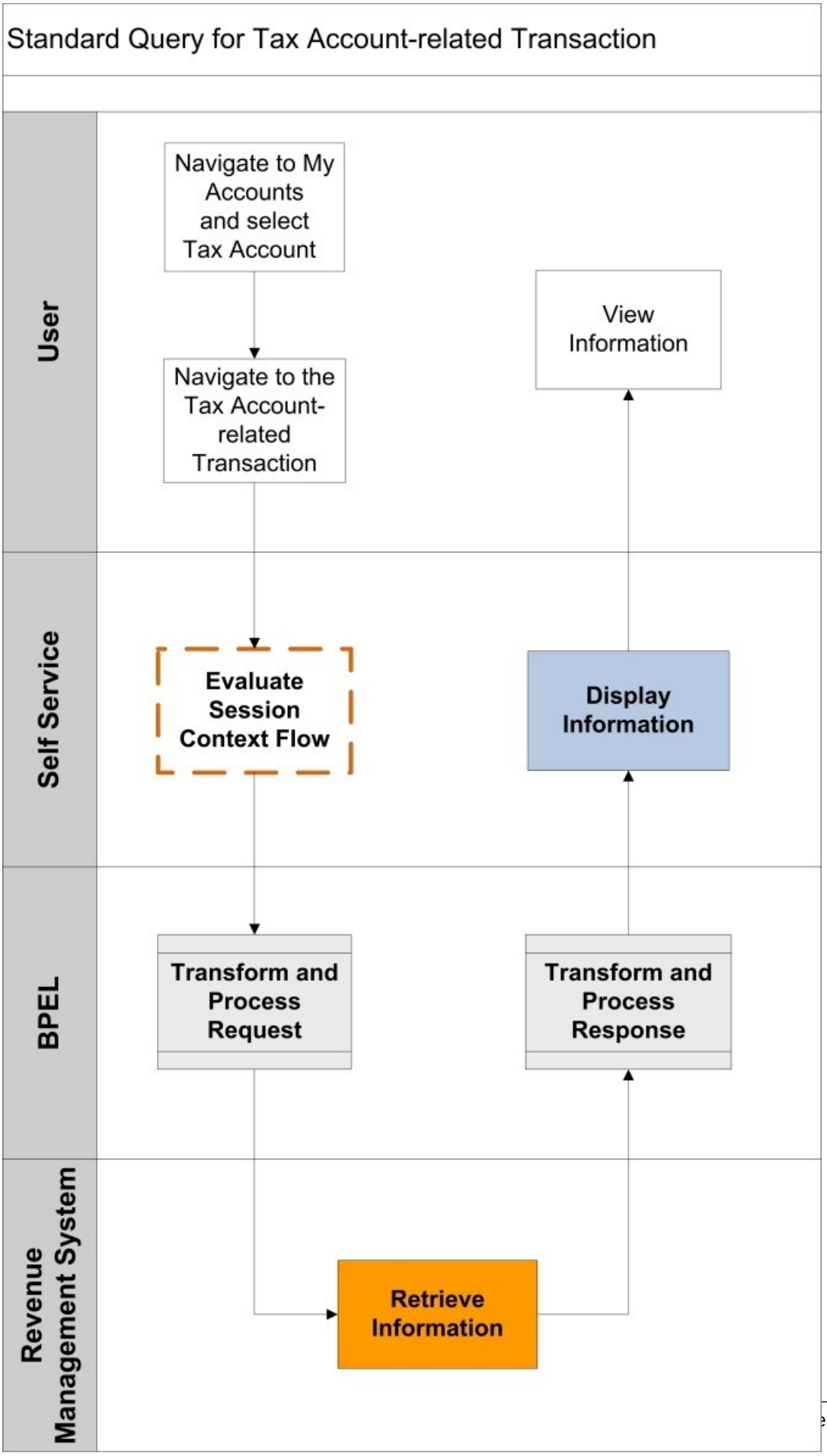


Figure 6: Standard method for retrieving Taxpayer Information

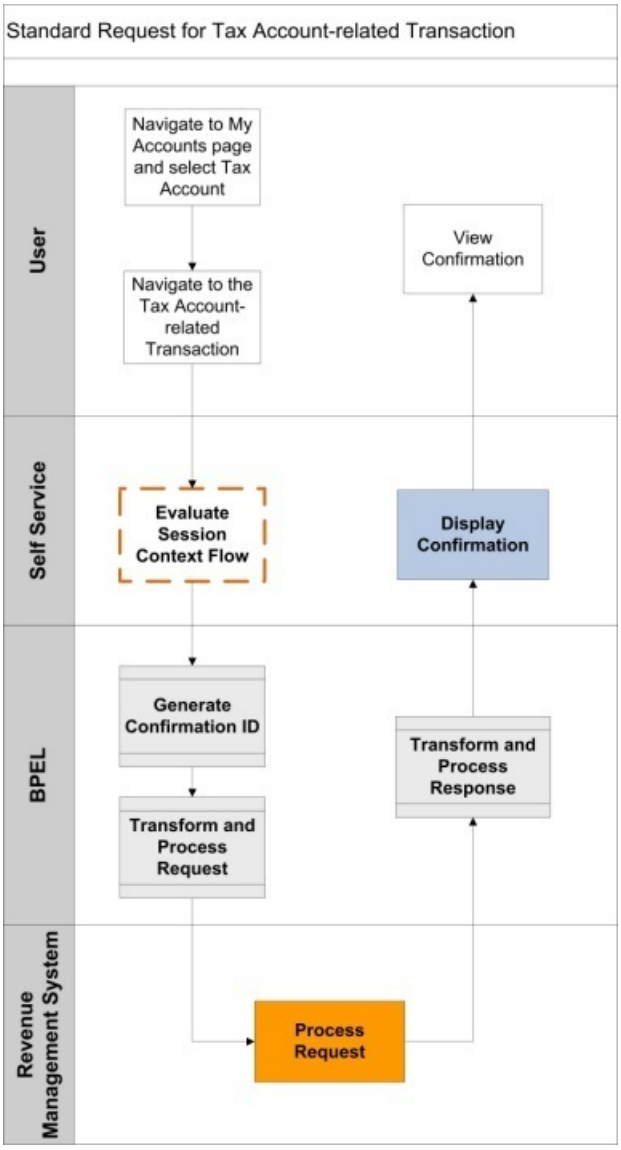


Figure 7: Standard method for updating Taxpayer Information

Taxpayer Summary

The Taxpayer Summary displays the information retrieved by the *Get Taxpayer Summary* web service.

The summary shows essential facts about the taxpayer, including primary contact information, if available.

The structure of the summary is as follows:

- **Title** – Short description of the taxpayer, such as name and primary ID.
- **Details** – The most important facts your implementation may wish to bring to the user's attention, e.g., taxpayer type's description (Corporation, Individual, etc).
- **Primary Contact** – Name and an email address of a primary contact.

Related Configurations

Taxpayer Info Configuration

The Taxpayer Summary display is controlled by the following configuration:

- **Taxpayer Summary Title Message** - this message is used to compose the Title. The *Get Taxpayer Summary* web service response contains Taxpayer Summary Title parameters. They will be used as Message substitution parameters.
- **Taxpayer Summary Details Message** - this message is used to compose the main text of the summary. The *Get Taxpayer Summary* web service response contains Taxpayer Summary Details parameters. They will be used as Message substitution parameters.
  Taxpayer Summary Title and Details parameter data types should be coordinated with the corresponding Message's parameters. If the parameter's data type is ***Lookup*** , the values returned by the web service should match one of the Lookup's Values.

The base product provides the following configurations:

Taxpayer Summary Title

The out-of-the-box solution applies Message 40001 to the taxpayer summary's title. The message text `'{1} - {2}'` contains two substitution parameters:

- 1- data type ***Text*** for **Taxpayer Name**
- 2 - data type ***Text*** for **Taxpayer ID**

For example, to display the title:

ABC Corporation, Ltd - XXXXXXXXX

the summary title parameters should be: **ABC Corporation** and **XXXXXXXXX**

Taxpayer Summary Details

The out-of-the-box solution applies Message 40002 to the taxpayer summary's details. The message text `Taxpayer Type: {1}` contains one substitution parameter, which is of data type ***Lookup,*** where the Lookup field is ***TAXPAYERTYPE***.

For each tax type used in the Revenue Management System, add a value to the Lookup ***TAXPAYERTYPE*** .

For example, for summary details to display:

Taxpayer Type: Corporation

the summary title parameters should be: **CORPORATION**.

In order to display Taxpayer Summary Details properly, the parameter value CORPORATION (description: Corporation) returned by the web service should exist as a value of the Lookup ***TAXPAYERTYPE***.

> **Note:** If your implementation wishes to modify the title and details' structure and/or use different parameters, you may override the message text and edit the message's parameters collection.

Contact Information

Contact Information displays the list of phone numbers and an email address retrieved by the *Get Taxpayer Contact Info* web service.

Each line on the phone list contains.

- **Phone Type** (see *Related Configurations*)
- **Phone Number and Extension**
- A single **Action** is available for Contact Information, which is **Edit**. This action opens a pop-up window allowing the user to add, delete or edit phone numbers and/or the email address. When the user presses the **Save** button, the *Get Taxpayer Contact Info* web service is invoked.

Upon successful processing of the contact information update request, the user receives a confirmation message containing the confirmation ID and details.

Related Configurations

The phone type selection is derived from the values of a Lookup, e.g., ***PHONETYPE*** .

The actual phone types (value of <phoneType> element) retrieved from the web service response are mapped in the Integration layer. See *Phone Type DVM* for more details.

Phone types can be added to the lookup and mapped according to the implementation's business requirements.

Correspondence Information

Correspondence information includes addresses associated with the taxpayer.

Addresses List

The **Get Correspondence Info** web service response includes a list of common Address XML fragments. Each Address is displayed according to the Address Configuration for its Country (value of the <country> element).

Each entry in the addresses list contains:

- **Address Usage** (see *Related Configurations* for additional information.)
- **Address Change Reason**
- **Address Data** - A common XML fragment that includes standard address fields.
- **Override Formatted Address Info**
- **Actions**
- **Edit** - Link that opens an **Edit Address** popup window.

If no correspondence information is returned by the web service, an **Add** action becomes available. It is displayed as a link, which in turn opens an **Edit Address** popup window.

Add/Edit Single Address

This window displays fields for a single address. When the user clicks **Save**, the *Maintain an Address* web service is invoked.

On successful processing of the request, the user receives a confirmation message containing the confirmation ID and details.

Appearance, visibility and validation of address fields is controlled by the Address Configuration (see *Related Configurations*).

Related Configurations

Address Label

The **Address** label (e.g., the Mailing Address) is derived from the value of the <addressUsage> element on the service. This value corresponds to one of the values of the ***ADDRESSTYPE*** Lookup. The value returned from the back-end system is translated in the integration layer using *Address Type DVM*.

Address Change Reason

The selection of Address Change Reasons that appear on the Edit Address popup is derived from the values of the ***ADDRCHANGEREASON*** Lookup. The values are mapped in the integration layer using the *Address Change Reason DVM*.

Address Configuration

Address Configuration controls the appearance of the address on screen.

Configuration is applicable for web services whose XML contains the common Location fragment.

The Location XML fragment includes:

```
<location>
<addressId/>
<addressUsage/>
<effectiveDates>
<startDate/>
<endDate/>
<seasonStart/>
<seasonEnd/>
</effectiveDates>
<address>
<name/>
<country>
<address1/>
<address2/>
<address3/>
<address4/>
<streetNumber1/>
<streetNumber2/>
```

```
<county/>
<city/>
<state/>
<postal/>
<houseType/>
<latitude/>
<longitude/>
<inCityLimit/>
<comments/>
</address>
<overrideFormattedAddress/>
</location>
```

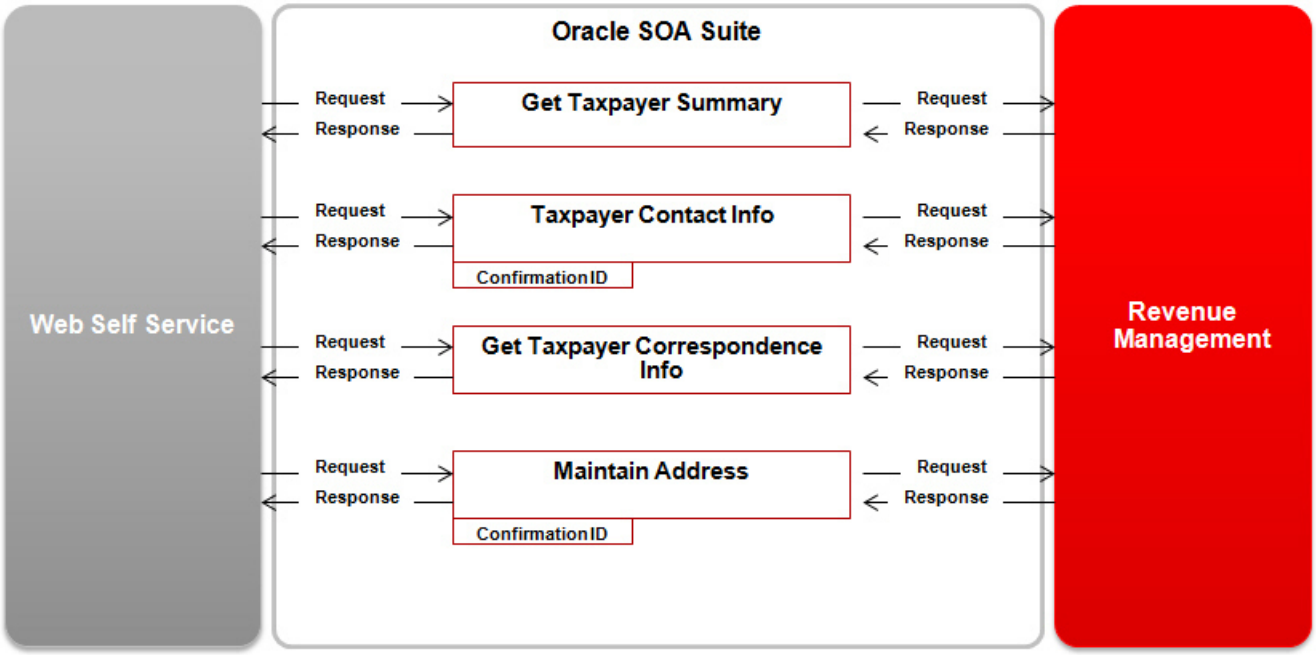
A non-customizable Lookup ***ADDRESSFIELD*** contains the list of all available elements of the fragment.

One Address Configuration is defined per Country. The base product sample configuration is delivered for Country Code USA.

See *Address Configuration* in the Settings and Configuration section of this document for detailed information on configuring addresses.

BPEL Processes

Integration Overview



The confirmation number is generated when Maintain Address and Taxpayer Contact Info requests are invoked. The confirmation number is returned as part of the response.

Implementing Taxpayer Information

This section describes the steps required in order to implement Taxpayer Information functionality.

Web Services

In order to support viewing of taxpayer information, the Revenue Management System is expected to implement the following web services:

Web Service	Description
TSGetTaxpayerSummary	This service retrieves a high-level overview of the taxpayer including taxpayer name, taxpayer type, primary contact details, and taxpayer summary. The request contains tax account identifiers (access keys +access type) and a line of business. The expected response includes: • **Summary Title** parameters list. • **Summary Details** parameters list. • **Taxpayer Name** (required, used internally by the self service). • **Taxpayer Type** (optional, used internally by the self service). • **Primary contact details**: contact name and email address. The information is expected to be derived for the taxpayer identified by the **Access Key 1** value.
TSGetTaxpayerContactInformation	This service retrieves a list of taxpayer's phones and an email address. The request contains tax account identifiers (access keys+access type) and a line of business. This request is submitted with two actions: Read and Update . For read, the expected response includes: • **Phones list**; each entry includes phone type, phone number and extension. • **Email address**. The information is expected to be derived for the taxpayer identified by the Access Key 1 value. For updates, the request contains an updated phones list and email address. The expected response should contain a confirmation message.
TSGetTaxpayerCorrespondenceInformation	This service retrieves a list of addresses associated with the taxpayer. The request contains tax account identifiers (access keys+access type) and a line of business.
TSAddressMaintenance	This service is used for add/update of a single address entry. The request contains tax account identifiers (access keys+access type), line of business, and address details, including an address change reason and address type. The expected response should contain a confirmation message.

Supported Address Configurations

The address configuration sample below is provided for reference purposes only and can be modified by the implementation.

Country	Description
USA	**United States of America**. This sample address configuration conforms with the most commonly used format for mailing addresses. The address fields include: • **Address line 1**. Required, visible, display sequence 1, parameter 1 • **Address line 2**. Optional, visible, display sequence 2, parameter 2 • **City**. Required, visible, display sequence 3, parameter 3 • **State**. Required, visible, display sequence 1, parameter 4 • **Postal**. Required, visible, display sequence 1, parameter 5 • **In City Limit** indicator. Optional, visible, display sequence 6 • **Country**. Required, visible, display sequence 7 • **Formatted Info Message** number 40010, message text: `{1} {2}, {3}, {4} {5}`

Messages

The following messages are defined to support Taxpayer Information functionality:

Message Number	Description
40001	`{1} - {2}`
40002	Taxpayer Type: `{1}`
40003	Address updated.
40004	Taxpayer Contact Information updated.
40009	We cannot process your request due to invalid phone format, please contact us for more details.

Configuration

In order to enable the supported service requests, you may add/modify the following configurations:

Address Configuration

Configure address for each country your implementation intends to support.

See *Address Configuration* in the Settings and Configuration section of this document for detailed information on configuring addresses.

Lookup

Add values for the following Lookups:

- **TAXPAYERTYPE**. Add entries for all taxpayer type codes (e.g., **IND**-Individual Income, **BUS**-Business, **CORP**-corporation) that your implementation may wish to expose on the self service portal.
- **PHONETYPE**. Add entries for all phone types used in your revenue management system that your implementation may wish to expose on the self service portal.

- **ADDRESSTYPE**. Add entries for each address type supported by the implementation.
- **ADDRCHANGEREASON**. Add entries according to your business requirements.

  Note: Translation using Domain Value Mapping is required if the lookup values configured above are different from those used in the revenue management system.

BPEL DVM Mapping

OTSS_AddressType

An entry should be provided for each Address Type included in the base product. The column ***OTSS_AddressType*** contains Address Type codes.

For each entry, specify the corresponding translation values from your Revenue Management system in the column ***EXT_AddressType*** .

If your implementation in the Revenue Management system is designed to use the same address types as in the web self service portal application, don't create any records from the DVM. As a result, the value in the <**addressType**> node on the request xml would be delivered 'as is'.

Examples:

The content of the node:

```
<addressType>MAILING</addressType>
```

- With translation value XXX for MAILING

```
<addressType>XXX</addressType>
```

- Without an entry in the DVM for MAILING

```
<addressType>MAILING</addressType>
```

OTSS_AddressChangeReason

The column ***OTSS_AddressChangeReason*** contains values of the Lookup ***ADDRCHANGEREASON*** .

For each entry, specify the corresponding translation values from your Revenue Management system in the column ***EXT_AddressChangeReason*** .

If your Revenue Management system does not use the address change reason, leave the translation value blank. As a result, the <**addressChangeReason**> node on the request xml would be empty.

If your implementation in the Revenue Management system is designed to use the same address change reasons as in the web self service portal application, delete the records from the DVM. As a result, the value in the <**addressChangeReason**> node on the request xml would be delivered 'as is'.

Examples:

The content of the node:

```
<addressChangeReason>RELOC</addressChangeReason>
```

- With translation value XXX for RELOC
- <**addressChangeReason**>XXX</**addressChangeReason**>
- With blank translation value for RELOC

<addressChangeReason></addressChangeReason>

- Without an entry in the DVM for RELOC
- <**addressChangeReason**>RELOC</**addressChangeReason**>

OTSS_PhoneType

An entry may be provided for each phone type. The column ***OTSS_PhoneType*** contains values of the .Lookup ***PHONETYPE***

For each entry, specify the corresponding translation values from your Revenue Management system in the column ***EXT_PhoneType*** .

If your Revenue Management system does not use the phone type, leave the translation value blank. As a result, the <**phoneType**> node on the request xml would be empty.

If your implementation in the Revenue Management system is designed to use the same phone types as in the web self service portal application, delete the records from the DVM. As a result, the value in the <**phoneType**> node on the request xml would be delivered 'as is'.

Examples:

The content of the node:

<phoneType>HOME**</phoneType>**

- With translation value XXX for HOME

<phoneType>XXX**</phoneType>**

- With blank translation value for HOME

<phoneType></phoneType>

- Without an entry in the DVM for HOME

<phoneType>HOME**</phoneType>**

OTSS_MessageNumbers

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

Chapter 8

Account Information Portal

Overview

The page is accessible to registered and enrolled self-service users. It consolidates the most important information about a tax account: *Account Alerts*, *Account Summary*, *Filing History* and *Payment History*.

All data is retrieved for the tax account currently in context. The description of the tax account is displayed in the top right corner of the main page area.

See *Enrollment* for more details about tax account definitions.

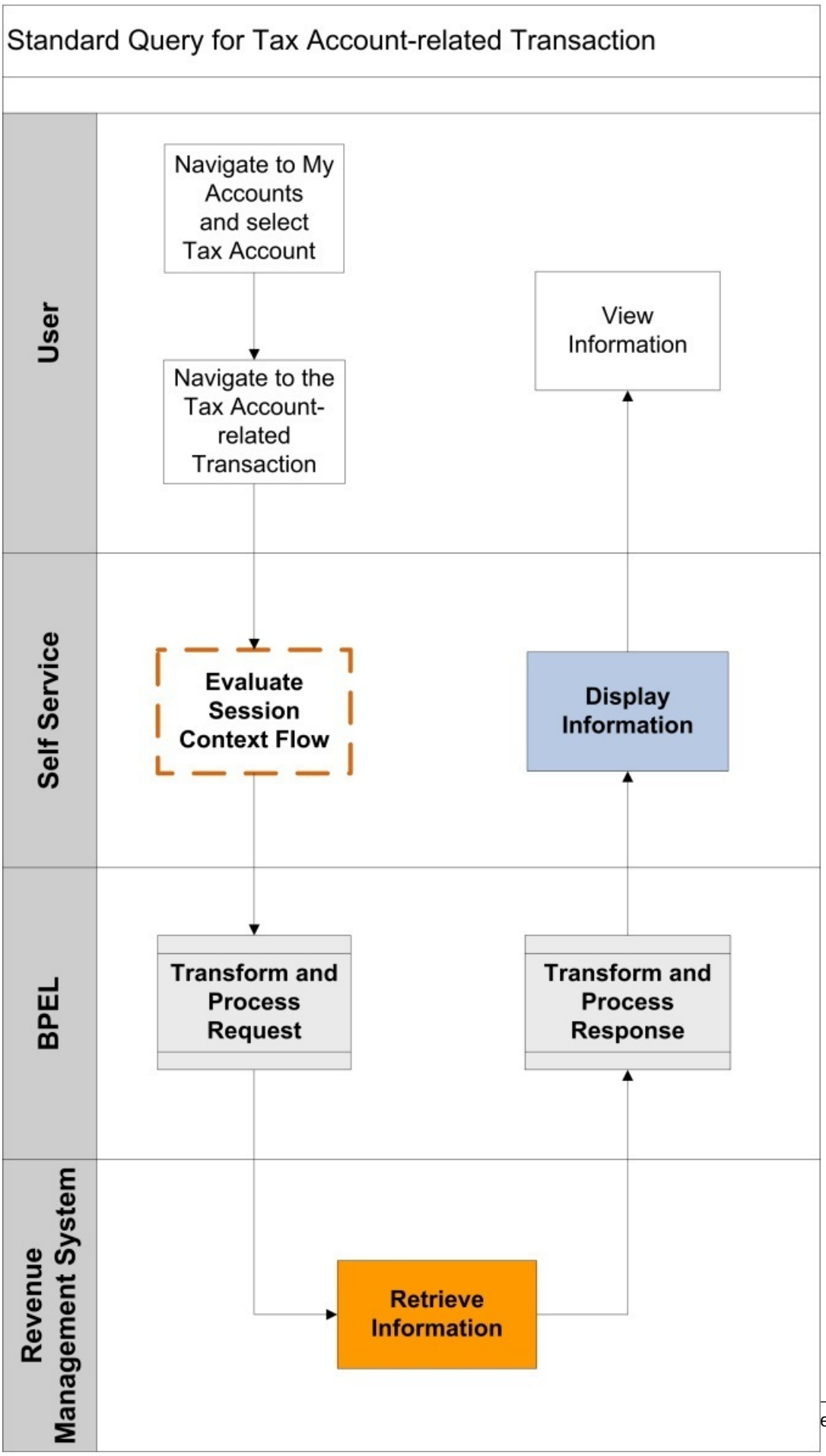


Figure 8: Standard method for retrieving Tax Account Information

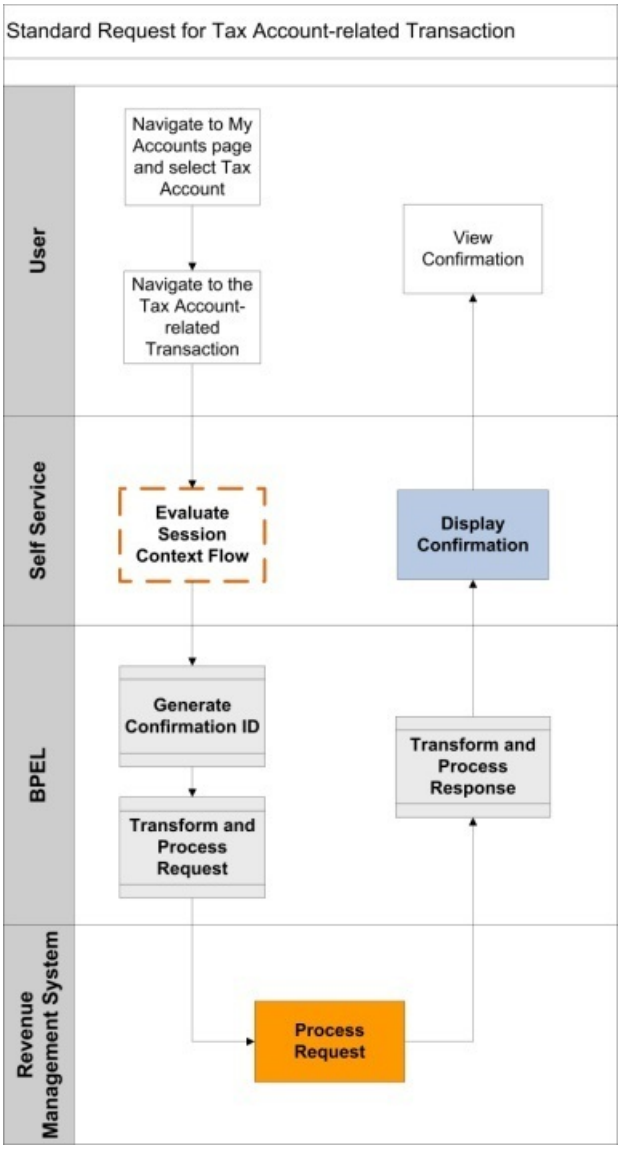


Figure 9: Standard method for updating Tax Account Information

Alerts

Alerts inform the taxpayer about various urgent matters related to the tax account and in some cases advise immediate action to resolve an outstanding issue. Alerts may also be used to communicate tax legislation changes that may affect the taxpayer or provide reminders about important tax-related dates.

The Alerts region displays two categories of alerts: the effective alerts for the current account:

Effective alerts for the current account. These alerts are based on the information maintained in the revenue management system and reflect the current taxpayer's and tax account's state. The list of alerts is retrieved by the *Get Tax Account Alerts* web service.

Alerts pre-configured in the self service application. These alerts contain "static" content, usually related to generic tax-related issues. They can be for announcements and reminders applicable for every self service portal visitor.

Alert displays the following:

- **Alert Title** - See *Related Configurations*.
- **Alert Text** - The message that is composed using *configurations* and the data returned by the web service response.
- **Icon** - An image illustrating the type of alert.
- **Navigation** link - The prompt for action related to the alert (see *Related Configurations*).

A single alert entry on a web service response contains:

- Alert Type
- Alert parameters list
- Document location (optional)
- Payment Amount (optional)

Alerts are displayed according to the rules defined on *Alert Type*. These rules define alert's contents, appearance and availability. In order to appear on the list, alert should be activated. Alerts can be made available permanently, during a specific time period, or seasonally (covering a certain time of year).

Message text configuration supports basic HTML and it can be used to emphasize certain parts of the alert's text. Alert parameters are used by the self service application as substitution parameters for the alert message. If the document location is included in the web service response it is displayed as a hyperlink.

Related Configurations

Alert's appearance, contents and related navigation options are controlled by the configurations defined on Alert Type.

Alert Type

To configure Alert Types, navigate to **Configurations > Alert Type.**

Information

Description is displayed as the Alert Title.

Active indicates if the alert is currently active.

Icon references a graphic file stored in the portal application's image repository. This is optional. The preferable size of the icon is 16x16 pixels. The image file should be added to the shared extension library (`/public_html/images`).

Alert Source defines the origins of the alert:

- **Revenue Management** source is used for **dynamic** alerts retrieved from the back-end system via web service. These alerts could be related to the current taxpayer or account-related conditions such as open collections or overdue balances.
- **Oracle Tax Self-Service** source is used for **static** alerts configured in the self-service application. These alerts allow the Revenue Management agency to deliver important information to portal users. **Static** alerts are displayed according to the **Effective Dates** setup.

For example, a static alert reminding users about individual tax return filing deadline could be made effective for a month prior to the deadline date.

Limitation: Messages used for static alerts should have no parameters.

Message represents the main alert text. Alert Parameters from the *web service* response are used as the message's substitution parameters.

When designing an alert, determine what information items needs to be retrieved by the web service. It may include amounts, dates and other important data.

The alert parameter's data type should be coordinated with the Alert Message's parameters. If the parameter's data type is ***Lookup*** , the values returned by the web service should match one of the Lookup values.

Example:

Assuming your Alert message should be:

`Your Individual Income Tax return is due on 04/15/2014.`

The Message text '`Your {1} return is due on {2}`' has two substitution parameters:

1 - data type ***Lookup*** , Lookup ***TAXTYPE***

2 - data type ***Date***

In order to display the alert properly, the alert parameter value returned by the web service should exist within values of the Lookup ***TAXTYPE.***

Effective Dates are applicable for **static** alerts (source **Oracle Tax Self Service**) and define the date range when the alert should be displayed. There are two methods to define effective dates:

Start and **End Dates** - define an absolute date range. This method could be used to alert the user about emergencies, new regulations and other one-time events. For example, an alert could be set to announce new tax credits/benefits.

Season Start and **End** - defines a season's boundaries. This method could be used for repeatable reminders, such as once-a-year individual tax return filing deadline.

Navigation

The following options control an alert's navigation properties:

Navigation Text is the text used for the navigation hyperlink.

Navigation Type defines the target transaction for navigation. Available options:

- **Payment** - Initiates a one-time payment flow. When using this navigation type, you should select an appropriate Payment Destination.
- **Service Request** - Initiates a service request. When using this navigation type, you should select the Service Request to use.
- **Tax Assistant** - Initiates the interactive tax assistant. When using this navigation type, you should select an Interview.
- **Portal Page** - Initiates navigation to one of the pages within the portal application. When using this navigation type, you should specify the target page in the **Navigation URL** field. The syntax is `/faces/taxpayerinfo`.
- **External Web Page** - Initiates navigation to any publicly-accessible web page. When using this navigation type, you should enter a full URL (e.g., http://www.google.com) in the **Navigation URL** field. The page opens in a new browser tab.

Account Summary

The Account Summary region/section provides at a glance an overview of the essential details about the tax account, including an address and current balance. If the balance is greater than zero, the **Make Payment** button is displayed. The self-service user may choose to make a payment, which triggers the one-time payment flow.

The structure of the summary is as follows:

- **Title** – A short description of the tax account.

- **Details** – Essentials about tax account status, for example a start and end date, the date and the amount of the last payment etc.
- **Location** – An address associated with the account.
- **Current Balance** – The outstanding balance including penalty and interest forecasted as of the current date.

Account Summary information is retrieved using the *Get Tax Account Summary* web service.

Related Configurations

Address Configuration

Address Configuration controls the appearance of the formatted address.

The *Get Account Summary* web service response includes standard XML fragment containing address elements. These fields are displayed according to Address Configuration for the input Country (value of the <country> element).

See *Address Configuration* in the Settings and Configuration section of this document for detailed information on configuring addresses.

Access Type

Access Type defines the scope of the tax account that is managed through self-service, controlling the appearance and content of the *Enrollment Summary* and the *Tax Account Summary*.

See *Access Type Configuration* in the Settings and Configuration section of this document for detailed information on Access Type configuration.

Filing History

Filing History displays the information retrieved by *Get Tax Account Filing History* web service.

Each line contains the data related to a single filing period. Items that are hidden and not displayed on the result grid: **amount due**, **printable form location URL**.

Filing History displays:

- **Filing Period** start and end date
- **Form** description. If the Form is defined in web self service and the value in <formType> element corresponds to an existing Form Definition, the description is displayed. Otherwise the value in <formType> is displayed 'as is'
- **Received Date**
- **Filing Due Date**
- **Filing Status** for the period
- Form's **Document Locator** - the identifier of the form assigned during submission
- **Confirmation Number** - for forms filed through web self service
- **Actions**

View Payment Submission Record - available if confirmation number is not blank. Triggers the navigation to *Track My Transaction* page with confirmation ID pre-populated

Related Configurations

The filing status displayed on Filing History line using the values of a Lookup ***FILINGSTATUS*** :

- **Complete** - for filing periods where all returns has been processed.
- **In Progress** - for filing periods where some processing is pending in the Revenue Management System.
- **File Now** - for filing periods where no returns has been filed yet.
- **Pay Now** - for filing periods where some payments are still pending.

The actual status codes (value of <status> element) retrieved with web service response are mapped in the Integration layer (see *BPEL DVM Mapping* for more details).

More statuses can be added to the lookup and mapped according to your implementation business requirements.

Payment History

Payment History displays the information retrieved by the *Get Tax Account Filing History* web service.

Each line contains the data related to a single payment:

- **Payment Date.**
- **Amount.**
- **Payment Method (**see *Related Configurations*).
- **Payment Status (**see *Related Configurations*).
- **Confirmation Number** - For payments submitted through web self service.
- **Actions.**
- **View Form Submission Record** - Available if the confirmation number is not blank. Triggers the navigation to the *Track My Transaction* page with a pre-populated confirmation ID.

Related Configurations

Payment Method

The payment method is displayed on a Payment History line using the values of a Lookup ***TENDERTYPE*** .

The actual status codes (value of <status> element) retrieved from the web service response are mapped in the Integration layer (see *BPEL DVM Mapping* for more details).

Payment Status

The payment status is displayed on a Payment History line using the values of a Lookup ***PAYSTATUS:***

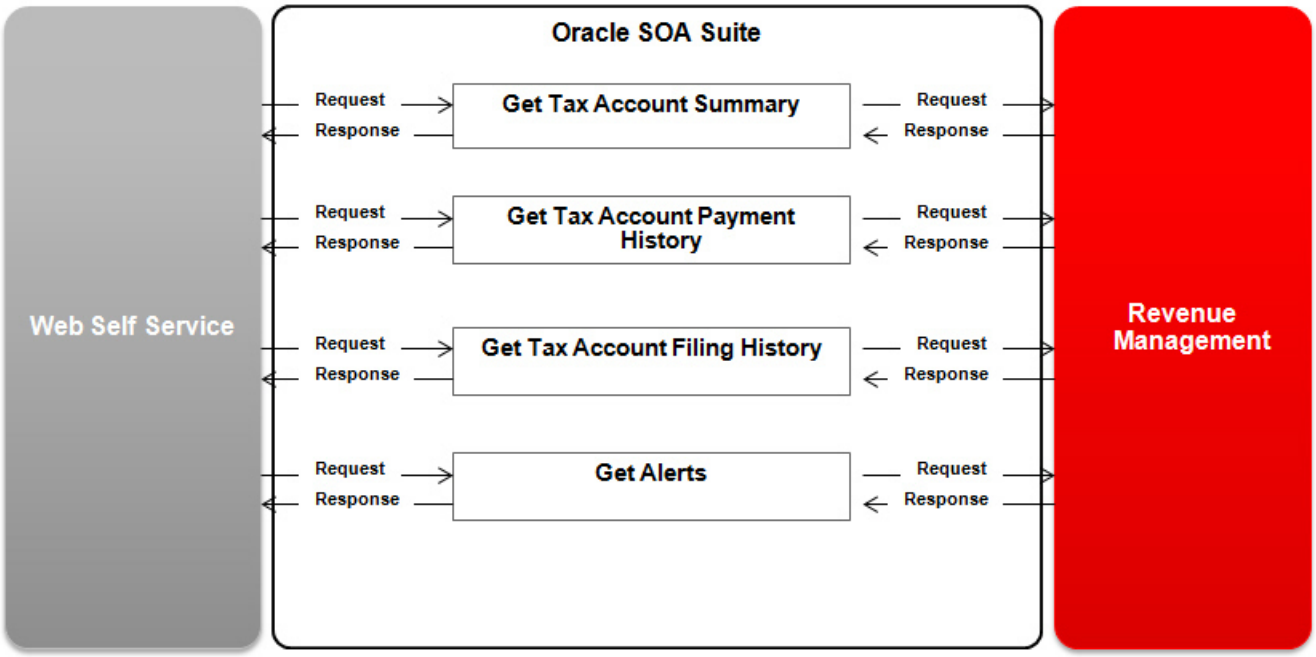
- **Complete** - for successfully processed payments
- **Issue Detected** - for payments that haven't been completed or ended up in error in the Revenue Management System.

The actual status codes (value of <status> element) retrieved from the web service response are mapped in the Integration layer (see *BPEL DVM Mapping* for more details).

More status values can be added to the lookup and mapped according to the implementation business requirements.

BPEL Processes

Integration Overview



The integration SOA composites invoke the Revenue Management System web services and send the response back to the self-service portal application.

Implementing Account Information

This section describes the steps required in order to implement account Information 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
TSGetTaxAccountAlerts	This service retrieves a list of alerts. The request contains tax account identifiers (access keys+access type) and a line of business. The expected response is a list of alerts, each entry includes: • Alert Type • Alert parameters collection • Amount due • Document location URL Request handling in the revenue management system is designed to work as follows: Retrieve account-related alerts and populate the output. Every entry on the output list should reference an alert type that is being translated in the integration layer into corresponding self service alert type.
TSGetTaxAccountSummary	This service retrieves a high-level overview of the tax account, including taxpayer name, tax type, current balance, account summary parameters, and an address associated with the account. The request contains tax account identifiers (access keys+access type) and a line of business. The expected response includes: • Summary Title parameters list. • Summary Details parameters list. • Taxpayer Name (required, used internally by the self service). • Tax Type (optional, used internally by the self service). • Current balance amount and currency (optional, used internally by the self service). • Address data (optional). Summary parameters expected to be derived according to the input access type. See *Supported Access Types* for more details.
TSGetFilingHistory	This service retrieves filing history for the input account. The request contains tax account identifiers (access keys+access type) and a line of business.
TSGetPaymentHistory	This service retrieves payment history for the input account. The request contains tax account identifiers (access keys+access type) and a line of business.

Supported Access Types

Access Type	Description
TAX_ROLE	Tax Account. This access type assumes that the tax account is identified by a single unique key. The access keys set for this access type include: • PER_ID – Alphanumeric taxpayer identifier in the revenue management system. • TAX_ROLE_ID – Alphanumeric tax account identifier in the revenue management system.

Access Type	Description

	The base product configurations referencing this access type are designed to support:
	• Tax account's start and end dates.
	• Location(s) associated with the tax account.
	• Ability to calculate and pay tax account balance.
	• Account summary:
	  • Title parameters tax type and a taxpayer name.
	  • Details parameters: tax role's start date, outstanding balance, last payment's date and amount.

Supported Alert Types

Alert Type	Description
OPEN COLLECTIONS	**Open collections**. This alert indicates that the tax account is associated with open collections. No alert parameters are expected to be returned by the web service. The web service response is expected to contain the amount due. This alert is configured to work with the payment destination Collection Notice (COLL_NOTICE).
OVERDUE BALANCE	**Overdue balance exists**. This alert indicates the existence of an overdue balance for the tax account. A single alert parameter containing the balance amount is expected to be returned by the web service. The web service response is expected to contain the amount due. This alert is configured to work with the payment destination Tax Account (TAX_ROLE).
RETURN MISSING	**Stop Filer Alert**. This alert indicates that a return is due for one of the filing periods. Two alert parameters are expected to be returned by the web service: tax type and filing period end date. The value of the tax type parameter should match one of the values of the lookup **TAXTYPE**. Alert navigation is configured to redirect the user to the online form filing page.
TAXPAYER INFO	**Taxpayer Info Incomplete**. This alert indicates that taxpayer record doesn't have a valid email address. No alert parameters are expected to be returned by the web service. Alert navigation is configured to redirect the user to the taxpayer info page.

Messages

The following messages are defined to support Account Information functionality:

Subject	Messages
Payment History	41010
Filing History	41010

Configuration

In order to enable the supported account information functionality, add/modify the following configurations:

Lookup

Add values for the following Lookups:

- **TAXTYPE**. Add entries for all tax type codes (e.g., **IND**-Individual Income, **SUSE**-Sales and Use, **VAT**-Value-Added Tax) that your implementation wishes to expose on the self service portal.
- **TAXPAYERTYPE**. Add entries for all taxpayer type codes (e.g., **IND**-Individual Income, **BUS**-Business, **CORP**-corporation) that your implementation wishes to expose on the self service portal.
- **TENDERTYPE**. Verify that all tender types used in your revenue management system have a corresponding value in this lookup. This is needed because the payment history list may include payments originating from sources other than self service.
- **STATE**. Add entries for the state attribute for the mailing address.
- **COUNTRY**. Add entries for all countries supported by the implementation.

  Note: Translation using Domain Value Mapping is required if the lookup values configured above are different from those used in the revenue management system.

BPEL DVM Mapping

OTSS_AlertType

An entry should be provided for each Alert Type included in the base product. The column ***OTSS_AlertType*** contains Alert Type codes.

For each entry, specify the corresponding translation values from your Revenue Management system in the column ***EXT_AlertType*** .

If your implementation in the Revenue Management system is designed to use the same alert types as in the web self service portal application, don't create any records from the DVM. As a result, the value in the **<alertType>** node on the request xml would be delivered 'as is'.

Examples:

The content of the node:

```
<alertType>COLLNOTE</alertType>
```

- With translation value XXX for COLLNOTE

```
<alertType>XXX</alertType>
```

- Without an entry in the DVM for COLLNOTE

```
<alertType>COLLNOTE</alertType>
```

OTSS_FilingStatus

An entry is provided for each filing status included in the base product. The column ***OTSS_FilingStatus*** contains values of the Lookup ***FILINGSTATUS*** .

For each entry, specify the corresponding translation values from your Revenue Management system in the column ***EXT_FilingStatus*** .

If your Revenue Management system does not use the filing status, leave the translation value blank. As a result, the <**status**> node on the request xml would be empty.

If your implementation in the Revenue Management system is designed to use the same filing statuses as in the web self service portal application, delete the records from the DVM. As a result the value in the <**status**> node on the request xml would be delivered 'as is'.

Examples:

The content of the node:

<filingStatus>PAYDUE**</filingStatus>**

- With translation value XXX for PAYDUE

<filingStatus>XXX**</filingStatus>**

- With blank translation value for PAYDUE

<filingStatus></filingStatus>

- Without an entry in the DVM for PAYDUE

<filingStatus>PAYMENTDUE**</filingStatus>**

OTSS_PaymentStatus

An entry is provided for each payment status included in the base product. The column ***OTSS_PaymentStatus*** contains values of the .Lookup ***PAYSTATUS*** .

For each entry, specify the corresponding translation values from your Revenue Management system in the column ***EXT_PaymentStatus*** .

If your Revenue Management system does not use the payment status, leave the translation value blank. As a result, the <**status**> node on the request xml would be empty.

If your implementation in the Revenue Management system is designed to use the same payment statuses as in the web self service portal application, delete the records from the DVM. As a result, the value in the <**status**> node on the request xml would be delivered 'as is'.

Examples:

The content of the node:

<status>COMPLETE**</status>**

- With translation value XXX for COMPLETE

<status>XXX**</status>**

- With blank translation value for COMPLETE

<status></status>

- Without an entry in the DVM for COMPLETE

<status>COMPLETE**</status>**

OTSS_PaymentType

See the *OTSS_PaymentType* topic in the BPEL DVM Mapping section for a description of this DVM.

OTSS_MessageNumbers

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

Chapter 9

Online Forms

The self service user is offered a selection of forms available for online filing. Once the form is completed, it is sent to the revenue management system for further processing. The form processing flow includes an ad-hoc identification step for the casual users and also a scenario where the user is a first-time filer.

The online form functionality provides form design capability and supports form submission via web services. The provided solution also includes document locator number generation logic and on-demand validation of the form data. The following information outlines the functionality at a high level.

Generic form data model. Form appearance and behavior vary from one form to another. However, the underlying data model is universal:

- The *form* contains one or more *sections*.
- The *section* contains multiple *lines* and/or *tables*.
- The *table* contains *rows*, each table row in turn includes one or more *fields*.

A nuclear unit carrying the actual form line's value is a *field*. Supported field data types include text, number, currency, boolean, date, date time, and lookup.

The XML representation of the form is self-descriptive; each <field> group element includes <value> and <dataType>.

Configurable forms. All aspects of the form filing are configured in the self service application. The configurations define the structure, the contents and the appearance of the form, processing rules and the post-submission activities such as upload of supporting documents and payment requirements.

Dynamic form rendering. The appearance of the form is controlled by the form definition. It may be amended through the configuration. The changes become effective immediately and do not require application re-deployment.

Filing Assistance. The form is equipped with various forms of online help. It includes:

- Interactive filing advisors, field-level tooltips, and section-level popup Help.
- HTML-enabled hints.
- Form instructions document retrieved from the Web Content Management server.

Interactive filing advisors are implemented using OPA Web Interviews. All filing assistance features are configurable and can be associated with the various elements of the form.

OPA-based validation. The information entered by the taxpayer is verified using OPA rulebases. This approach allows minimizing manual data entry, preventing calculation errors and overall improving the accuracy of the filing. The integration is possible based on the following premises:

- The self service application operates with metadata-driven, generic form data model.
- OPA supports generic API that allows invoking of any rulebase, provided the rulebase data structure is "known" to the caller.

Configuration control. The product allows self service application administrator to restrict user's ability to modify form definitions and enable forms for online use.

Configuration import. Form definitions may be imported from another system.

Form Design

Form Designer

Form Designer feature provides a facility for online form configuration. The following can be configured using the designer:

- Form structure and data model.
- Form appearance and in-line help, including interactive filing advisors.
- Sensitive data protection with data masking.
- Identification requirements. Whether the taxpayer should be identified in order to be allowed to file.
- Form's availability for online filing.
- Orchestration of the form submission flow and a subsequent payment and supporting file upload sub-flows.
- Applicability of the various form's actions: form data validation by the back-end system, copy form data from the previous return, and custom-defined action(s).
- Form validation rules.

The feature allows implementers to define the form in multiple languages and release one language at a time.

Once the design work is completed, the system combines all the information needed for dynamic rendering in a single XML file. This "compiled" XML layout is stored in a database table and used at run-time. Authorized users are allowed to enable the form for online use.

The following steps should be taken to configure a form:

- Create or import a form configuration.
- Design the layout and fine-tune form's appearance.
- Determine the requirements for interactive assistance and in-line help based on the complexity of the form.
- Design interactive Filing Advisors and link them to the appropriate form elements.
- Define identification, payment and supporting documents upload requirements.
- Create OPA rule for form data validation and link it to the form definition.
- Finalize the layout and enable the form for online use.

Select **Configurations** > **Form Designer** to define a form's general information, online availability and identification options.

Defining General Behavior

Navigate to the **Main** tab to define the form's general behavior.

General Information

Form Type code is the main identifier for the form across the system. It is recommended that this identifier be meaningful and correspond to a real form name. For example, ***1044IND*** for the U.S. ***1040 Individual Income Tax Form*** .

Description appears on various queries and pages where the form type is referenced.

Category is a broad category for form types. There are two supported categories: ***Tax Form*** and ***Business Registration Form*** . Each category is processed by a dedicated web service and has its own self-service page that lists the available forms.

Currency specifies the currency that applies to the form type. If not specified, the form is rendered and processed using the default system currency (for the current currency setting, see Admin, Settings > System Configuration Options).

Filing Advisor is an interactive interview that can be launched directly from a list of available forms. Search allows selecting interviews from Interview Sets of a category ***Form Filing Advisor*** . For example, the form-level Filing Advisor may help determining whether or not the user should file this particular form. If a Filing Advisor is specified, the user can access the specified Interview from two locations: by clicking an information icon next to the form on the list of available forms, and from a link on the form itself.

See the *Oracle Policy Automation* section for detailed instructions on creating OPA web determinations and configuring the Filing Advisor.

Form Instruction is displayed on the collapsible panel on the right side of the online form.

To display the form instruction, create an HTML file and upload it to the Web Content Management server connected to the portal. Make sure the content item is accessible externally.

As an example, the following UCM document properties can be used to allow public consumption:

- Security Group: Public
- Account: <blank>

Populate the **Form Instruction** with the newly-created content's ID using the following syntax:

```
UCM#dDocName:TS_FORM_INSTR_SAMPLE
```

> **Note:** Also see the *UCM documentation* for detailed information about general security considerations and document security.

Online Availability

The form's online availability is defined in one of two ways:

- Specify **Start** and **End Dates** to define the period in which the form type could be filed. Outside of this date range, the form type will not be displayed on the list of available forms.
- Specify **Blackout Period** and **Message** to define the time of the year when the form type is not available and the message that will be displayed if users make an attempt to file forms of the type.

Identification

Identification Required? indicates if the user should be identified by the back-end system in order to file the form. An option to search for an appropriate **Identification Request** is provided. For additional information see *Taxpayer Identification Request*.

If **First-Time Filers** are allowed, the back-end system is expected to check the validity of the identification details, but not necessarily taxpayer's record existence. If no errors and no ID-s are returned by Identification Request, user may proceed further with the filing.

Defining Processing Options

Navigate to the **Processing** tab to define submission, post-submission and confirmation options.

Submission

Response Mode indicates whether the form should be processed synchronously or placed in the queue.

Post-Submission

The options in this section control whether or not the user is prompted for a form payment and/or supporting documents after the form is submitted.

Payment Required? indicates if payment step should follow successful form submission.

- If set to ***true*** , the user is prompted to pay and payment information is displayed on the form submission confirmation page.
- Set the indicator to ***false*** if a prompt for form payment is not needed, e.g., if the payment's details are incorporated into the form itself.

Flat Fee Amount could be defined on the forms used for business license application or one-time request for a certificate.

Alternatively, one of the form's fields could be marked as *Payment Amount Field* (for details, see the **Line** section in *Designing Form Layout*.

If both **Flat Fee Amount** and *Payment Amount Field* are configured, the system will check if the **Payment Amount Field** is populated; if so, it will use the value of the Payment Amount Field.

Override Payment Destination specifies the **PaymentDestination** of a special category of ***Form Payment.*** If populated, it will be is used to handle payment immediately following form submission. At the time of payment, the form may still be en route to the backend application (asynchronous submission) and the only identifier available would be the document locator number generated during submission.

The system expects this Override Payment Destination to include at least one field and populates this field with Document Locator Number.

If payment is required and no override payment destination is specified, the system will use payment destination specified on *Form Process Control*.

Payment Destination TAX_FORM is supplied out of the box.

Payment Instruction is text/HTML content that is displayed on the payment area of the form submission confirmation page.

Upload Supporting Document indicates if additional documents (e.g., scanned receipts, photos or other files) should be supplied with the form. If set to ***true*** , the user is prompted to upload these documents on the form submission confirmation page.

Upload Instructions is text/HTML content that is displayed on document upload region of the form submission confirmation page. File types allowed for upload are listed in *System Configuration Option* **UPL_FILE_TYPES**.

Confirmation

Confirmation Email

Specify an **Email Template** that defines email to be sent to the user upon successful form submission. A newly created Form Definition is populated with the template defined on *Form Category Process Control*. This default value can be replaced with another email definition. The email body is expected to contain the **{message_details}** substitution placeholder.

```
Dear taxpayer,

This is a confirmation of your form submission.

{message_details}

Superior Department of Revenue

101 Main Street, P.O. Box 0000-0000

Superior, SP
```

Message defines the text to be injected into a standard email body.

The system expects the message to contain four parameters: 1 - Form Description, 2 - Date Time, 3 - Document Locator, 4 - Confirmation Number.

Message 51301 is supplied out-of-box:

```
Form {1} was submitted successfully on {2}. Use Document Locator {3} and/or confirmation
number {4} for all your further inquiries
```

Confirmation Page

The message defined on the Form Definition is also displayed on the confirmation page.

Defining Actions

Navigate to the **Actions** tab to configure actions available for the form during online filing.

Form Actions

The **Form Actions** tab displays all active values of a lookup **FORM_ACTION**.

For every enabled (checked) Action, an action button is available during online form filing. Several actions are required and are available for every form at runtime. Clicking the action button invokes the **TSProcessTaxForm** web service with the <action> element populated with the lookup value.

Some actions are optional and can be disabled. Use the checkbox to disable/enable optional actions. The button's label is derived from the lookup value's description. It can be customized using the Lookup Value override description.

Required actions:

- **Edit** - Reopens online form for edit when form is ready to be submitted.
- **Ready To File** - Triggers action **READY** which is a last validation round before final submission. If this action returns no errors/exceptions, the form can no longer be edited.
- **Submit** - Triggers action **SUBMIT** which is sending a finalized form to the back-end system for the processing/posting; in response the service returns confirmation info.

Optional actions:

- **Check Form Data** - Triggers the action **VALIDATE** (see *Validation* for details).
- **Copy From Previous Return** - Triggers the action **COPY**. This action is expected to pre-populate the form based on the previous filings.
- **All custom actions**.

Custom Actions

Your implementation may wish to support additional actions during online filing. New actions can be added to the form processing with the following steps:

1. Extend the base SOA Composite and implement the new logic within the integration layer and/or the back-end system. Make it conditional, based upon the value of request's <action> element.
2. Add new active value(s)s to the FORM_ACTION lookup:
   - Lookup value is the value expected to be populated on the web service request's <action> element.
   - Description is the label for the form action button.
   - Extended value should be set to OPTIONAL or REQUIRED.
3. Enable new action on the appropriate Form Definition(s).

Custom Properties

Custom Properties may be used to categorize/group logically related forms; the form definition can be searched by a Custom Property code and value combination.

A Custom Property could also be included in Process Form web service request along with the form data.

For example, a custom property on the form can capture the city name. In the agency that serves multiple jurisdictions, this property can be used to group forms by city. It can also be used by the integration layer for city-specific message handling.

Defining Validation Rules

Form data is validated on several levels:

- **Single-line level**. The value is verified according to the line's data type. In addition, it may be validated using a regular expression. The Validation Rule is configured using the Layout Designer. See *Line* for more details.
- **Form-level validation by the back-end system**. If the back-end system allows ad-hoc validation of the form data, it should be indicated on the *Form Process Control*. Turn off ***Check Form Data*** action if no validation should be performed for a specific form.
- **Form-level validation using OPA Rules**. OPA-based Validation Rule can be associated with the Form Definition.

The form will be validated with an OPA rulebase for every form action, except for **Submit** and **Copy From Previous Return**.

To set the Validation Rule, navigate to the **Validation** tab and select Validation Rule from the dropdown.

Creating OPA Rulebase for Form Validation

The steps to create the OPA Rulebase for specific form validation are as follows:

- Press **Generate Rulebase Model** button and save the data model file locally.
- Change the file name's extension to .xsrc.
- Use this file as a properties file for the new Rulebase.

See *Form Validation Rulebase* for detailed development recommendations.

- Deploy the rulebase on the OPA server.
- Configure a new or use an existing *Interview Set Search* for Form Rule validation and add a new Interview.
- Create a new OPA-based *Creating a Validation Rule Using OPA*.

The new Validation Rule can then be linked to the form definition.

Designing Form Layout

The form's data structure, UI "look and feel", inline Help and filing assistance, and certain validation and processing rules are all defined using the Layout Designer.

Click the **Go To Layout Designer** link in the action buttons area (upper right side) of the Form Designer page.

On the Layout Designer page, the form's structure is displayed as a tree on the left-side panel. Click a node to view/edit its properties.

Form

Form is the root element for the entire structure. It is the first node you have to create.

Section

Appearance and Data Structure

Each section appears on a separate tab. It could contain a logically related group of input fields or represent a sub-form (e.g., Schedule).

The following attributes define a section's appearance:

- **Name** works as Section's primary identifier. It must be unique within the form. When a new Section is added, **Name** and **Label** are populated with default values that should be modified.
- **Display Sequence** controls the order in which the sections appear on the page.
- **Label** is displayed at runtime as a tab name.
- **Occurrence** defines if the section is repeatable.

Section-level Help

Filing Advisor is an interview that provides the user with interactive assistance and explanation about the section. If specified, the link is displayed on the top right corner of the tab. The link text is derived from the interview's description. Search allows selecting interviews from Interview Sets of a category ***Form Filing Advisor.***

Help - Static text/HTML help associated with the section. If specified, a help icon appears on the top right corner of the tab. Clicking on the Help icon displays the pop-up Help text.

Footer and **Header** are displayed above and below the section's fields. They may be entered as text or HTML.

Line Appearance and Data Structure

Name works as Line's primary identifier. No spaces are allowed. **Name** has to be unique within the section. When new Line is added, **Name** and **Label** are populated with default values that should be modified.

Label is displayed at runtime.

Display Sequence defines the position of the line on the display.

Line Number is an optional label. If specified, it will appear on the left of the line's label.

Field Reference (indicator and the actual Field).

A Field Reference is used to derive a line's properties - data type, length and precision, label, and validation rule from an existing Field.

Search for the Field or switch to the Component Tab of the Layout Designer and double-click on the Field. The prompt offers two options: reference the Field or just copy the Field's definitions.

- If Field is referenced on the Line, the Field's definitions are used for form rendering. Referenced Field could be modified and the changes are propagated everywhere by refreshing affected Form layouts (see *Reopening a Finalized Form* for details). This option is suitable for universally reusable fields with same label and validation rules applicable everywhere (e.g., for Email Addresses).
- If Field definitions are copied, they are editable and independently "own" by the Line. In this case Field is used as a template for line definitions. This option is suitable for Fields such as Name or Amount whose data type/size/length will remain the same system-wide, while label and validation rules may vary from one place to another:

When **Line** references a **Field**, the data type, length, precision, label, and validation rule are not editable.

Data Type is required. For data type Lookup, the **Lookup** name must be selected.

Data Length defines the maximum number of input characters.

Precision is required for numeric data types.

Predefined Value is a default value for the line and is automatically populated on a new form.

Display Only indicates that line is not editable. This option can be used for calculated lines whose value is determined by an OPA Rule (Form Validation Engine). It can also be used in conjunction with Predefined Value for displaying static information such as Tax Rate.

Payment Amount Line marks the Line that contains calculated payment amount for the form. Only one line can be selected. At runtime, the value from this line would be used for post-submission form payment.

Required indicates that Line must have a value populated. If the data type is numeric, user will be asked to enter 0.

Visible indicates whether or not the line is visible to the self-service user. By default, the Line with this setting turned on (e.g., visible). Certain fields containing static or internal information can be marked as not visible so the data is available to OPA Form Validation Engine and/or the back-end system, but hidden from the self-service user.

Masked indicates whether or not the Line's input value will be masked on screen.

Validation

Specify **Validation Rule** to validate line data using regular expressions.

Line-level Help

Hint is text that is displayed on screen beneath the line's label and the input widget. Hint may be entered as plain text or HTML.

Help is text displayed in a popup when the cursor is hovering over the input widget.

Filing Advisor is an interactive interview that may guide the user through complicated filing questions. If specified, the link is displayed next to the line's label. The link text is derived from the interview's description. A search allows for selecting an interview from Interview Sets of a category. ***Form Filing Advisor.*** For example, a line-level advisor may help to determine whom the taxpayer may claim as a dependent.

Table

Appearance and Data Structure

One or more Tables can be defined within the Section. The number of columns is not limited but screen dimensions should be considered at design time.

Each table row is displayed as a collapsible sub-form.

Name works as a Table's primary identifier. No spaces are allowed. Name must be unique within the section. When new Table is added, **Name** and **Label** are populated with default values that should be modified.

Display Sequence controls the position of the table on the screen.

Label is displayed above the first table row region.

Line Number is an alternative label, displayed on the left from the main label.

Table-level Help

Hint is plain text. If specified, it is displayed below the Table's label.

Filing Advisor is an interactive interview that may guide the user through complicated filing questions. If specified, the hyperlink is displayed next to the table's label. The hyperlink text is derived from the interview's description. A search allows for selecting an interview from Interview Sets of a category. ***Form Filing Advisor.***

Table Column

The configuration options for the column are essentially the same as for the regular *Line*, except for **Hint**, which is not applicable.

Previewing the Form

Form layout rendering may be tested through the Preview functionality. Navigate to Layout Designer and press the **Preview** button. To return to the Layout Designer, click the corresponding link.

Note: Preview should be used for the review of form rendering only; the form cannot be submitted from the preview screen and action buttons won't trigger any process;

Multi-language Support

When a new form definition is added, all language-based configurations are created in the current user's language.

In order to make the form available in additional language(s):

- Navigate to **Configurations > Form Designer** and search for an existing form definition.
- Highlight the entry on the search result grid and press the **New Language** button.

Finalizing the Layout

Completing the Design

A form could be made available, one language at the time. For example, the form could be defined in the most commonly used language and enabled for online use, while implementers continue working on additional languages, add and translate validation rules etc.

When the design is completed, a single language layout should be finalized.

The **Finalize Layout** button is visible on the Form Definition page if the user is authorized for that action.

Press the **Finalize Layout** button to create a version of the form in the currently used language.

Once the layout is finalized, the form could be *enabled for online use*.

When at least one language layout is finalized, the Form Definition cannot be deleted. *Support for additional languages* can be added and changes could be made to the finalized layout. In order for any changes to take effect, the layout should be finalized again.

Locking the Design

Once all the design activities are completed, the form may be locked in order to prevent any further modifications.

When locked, neither form structure nor form layout could be modified, no new languages could be added to the form definition

The **Lock** action button is visible on the Form Definition page if the user is authorized for that action.

.

Re-opening a Finalized Form

Unlock Layout for Re-edit

A non-closed form definition can be re-opened for edit if needed. Once unlocked, the form's data structure and layout can be modified - i.e. sections and/or fields can be added/removed/re-arranged. The language-based information can also be modified. New language support can be added.

IMPORTANT: When all the changes are completed, the form layout(s) should be **finalized** and new layout versions *enabled*.

The **Unlock** action button is visible on the Form Definition page if the user is authorized for that action.

Closing the Form

When the form is no longer in use the form definition may be marked as Closed. This action is an equivalent to the deletion of the form. Closed form definition cannot be reopened for edit and remains in the system for audit purposes only.

Form Designer Security Setup

In order to have the ability to **Lock**, **Unlock**, **Close**, and **Re-Import** a Form Definition, the user should belong to the enterprise application role **PSRMSSFormsAdmin**.

Import Form Definition

Implementations may import form definitions from another system that uses metadata-driven dynamic rendering.

Initial Form Import

To import a form definition, navigate to **Configurations > Import Form Definition** and search for forms available for import. The *Retrieve Active Form Types* web service is used to derive the list of forms eligible for import.

From the search results, select the form(s) to import. Multiple forms can be imported at once. Click on the **Import** button to proceed with the import.

Imported form definition can be further modified in the self-service portal application.

Import Form

The *Import Form Definition* web service is used to derive form definitions from the back-end system. It allows importing form structure (sections, lines, and tables), data types, field size and other attributes describing each field's data, labels and help text in multiple languages.

```
Example: information imported for a single line, in two languages: ENG and XXX
<line>
<displaySequence>1</displaySequence >
<field>
<name>totalAmount</name>
<lineNumber>12a</lineNumber >
<dataType></dataType >
<lookupName></lookupName>
<fieldLength>18</fieldLength >
<fieldScale>2</fieldScale >
<description>
  <languages>
    <text>Total Amount</text>
    <language>ENG</language>
  </languages>
  <languages>
    <text>XTotal XAMount</text>
    <language>XXX</language>
  </languages>
</description>
<help>
  <languages>
    <text>Total Taxable Amount</text>
    <language>ENG</language>
  </languages>
  <languages>
    <text>XTotal XTaxable XAmount</text>
    <language>XXX</language>
  </languages>
</help>
</field>
</line>
```

A field may be imported with an associated list of valid values. A new Lookup is created for each of these fields, with Lookup Name = Field Name and Lookup values representing each of the valid values imported. On the Form Definition, the field's data type is set as Lookup and the newly created Lookup is referenced.

When a form is imported from another system, the definition and lookups are created with owner ***Revenue Management***.

Form Field - Data Types

Each input field's data type should be configured in order to render and process the form correctly. Form Import may not contain sufficient information about the data. For example, it may contain a data type (Number, Date, Boolean etc) but may not specify the size or precision.

In those cases, the Form Field is created with a default 'placeholder' field reference. Use *Form Data Type* to define which Field to use for every available data type.

Refresh Lookup

The *Refresh Lookup* web service is used to retrieve an updated list of lookup values. New values are added in ***Active*** status. Values that aren't present in the returned list are deactivated.

In order to refresh a specific Lookup navigate to **Settings -> Lookup**, find the Lookup using a main search and then press **Reload Values** button on top of lookup Values grid.

Redesign Imported Form

The raw definitions imported from the back-end system may not render perfectly. They can, however, be enhanced with various on-screen help and other layout adjustments.

> **Note:** The structure of the imported form definition cannot be changed. Node names are protected and cannot be deleted. New nodes cannot be added. You can attach valid values list to the form line, hide internal information, define default values, add help, hints and interactive filing advisors, and also design form processing and post-submission activities.

Add Value Lists

A list of valid values can be added to a Line definition if the original data type is ***text*** . The steps are as follows:

- Create/find a Lookup that represents the values.
- Set Line's data type to Lookup.

For example, a Line represents a tax rate and in the back-end system it is defined as a Foreign Key to the Tax Rate table. For a particular form, you can create a Lookup that contains only applicable rates and modify the Line to reference this Lookup

Hide Internal Information

Hide lines that aren't applicable for online user by turning off the **Visible** setting. Mask Lines containing sensitive data items.

Edit Form Data

Determine and mark the Line that holds the Payment Amount.

Identify calculated Lines and mark them as Display-Only.

Add line Validation Rules.

Add Online Help

Section-level and Line-level help could be a part of the import. Imported text can be edited using Layout Designer. Basic HTML is supported.

Filing Advisors on all levels, Hints, Headers and Footers can be added using Layout Designer.

Define Form Processing and Submission

Configure the form's *General Behavior*, *Submission* and post-submission activities, including confirmation, payment and supporting documents upload requirements.

Form Re-import

In order to synchronize a self-service form definition with that in the back-end system, the form could be re-imported.

Re-import is a total replacement of the existing definition and all modifications would be lost, including hints, help, and other language-based configurations.

The **Re-Import** action button appears on the result grid of Form Definition Main Search and would be visible to authorized users

Enabling a Form for Online Use

Finalized form layouts (one layout per language) need to be activated for online use.

To enable a form for online use, navigate to **Configurations > Enable Forms**, select the Form Definition and search for available layout versions. Pick up the latest version in each language and activate that version.

Only one form layout for any given language could be active at any point in time. Old (deactivated) form layouts are kept in the system for audit purposes.

Security considerations

Enabling is the last step before the form becomes available for online use. This page can be secured separately and the access to it granted to a limited number of responsible users.

Form Data Type

Form Data Type stores a collection of Fields whose properties are used to default Form Line properties if no appropriate definition is available from the Form import.

Select **Configurations > Data Type** to maintain data types.

Data Type references one of the values of the ***DATATYPE*** lookup

Template Field defines a **Field** that represents this data type.

Form Validation

All forms are different and so are their validation and calculation rules. However, all forms in the self service application share the same data structure, and this structure is captured on the **Form Definition**.

The self-service application and OPA rulebase may communicate form data with each other as long as the rulebase is built on the data model produced from the Form Definition.

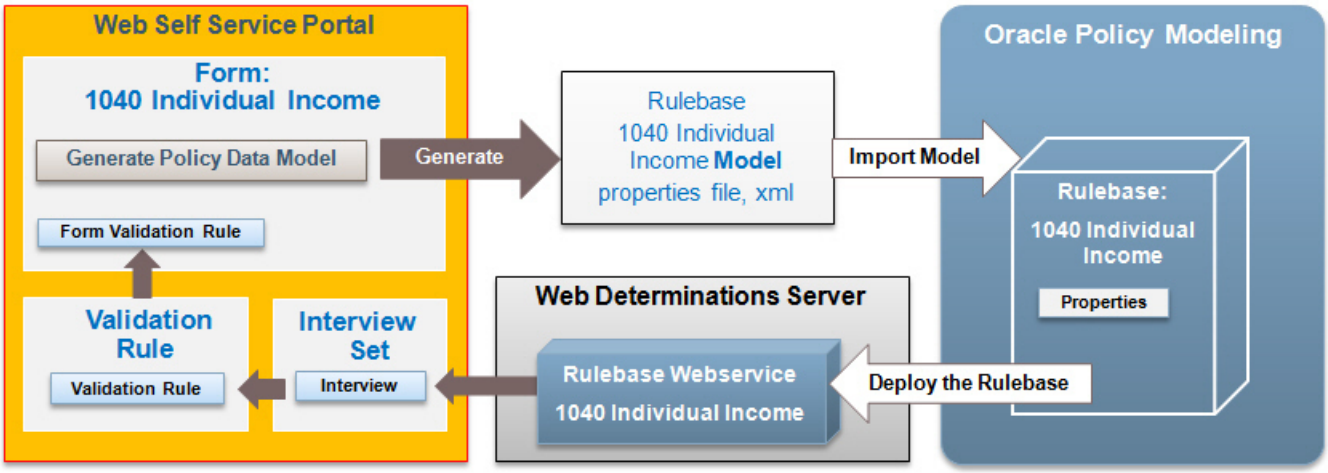
The generated rulebase data model contains entities representing the form's Sections, Lines, and Tables. The public name of the entity, along with its attributes, properties, and containment relationships, are derived from the **Form Definition**.

The automatic interaction with OPA rulebases is designed based on the following assumptions:

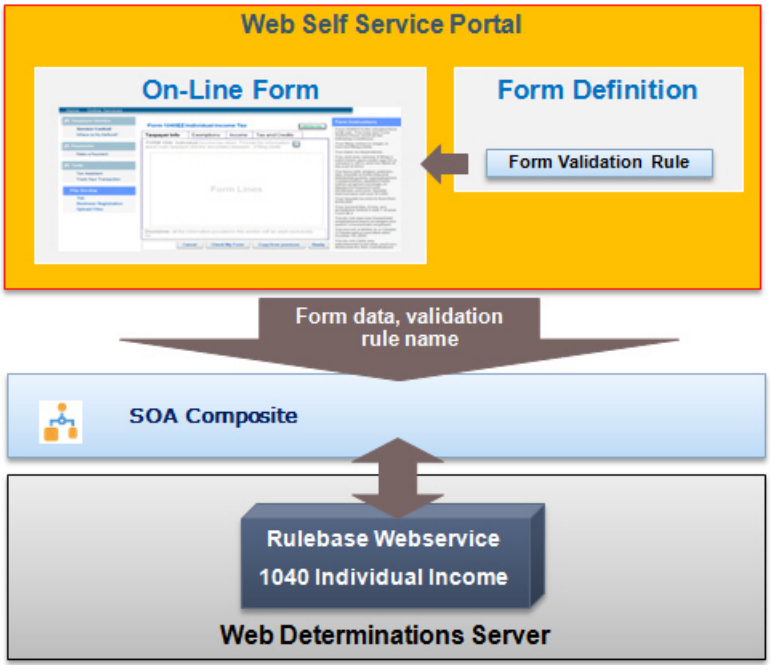
- The rulebase is built using the generated data model.
- Every rulebase may be invoked via web service using a single "generic" WSDL.
- The rulebase logic must cater for handling a partially-filled form. By default, the input for the rulebase is the entire form data. However, the self service user may trigger the validation, for intermediate calculation/verification purposes, before completing the filing, so the rulebase needs to handle incomplete information.
- The expected rulebase outcome is:
  - Form-level indicators of errors and updates.
  - Section-level indicators of errors and updates.
  - Table-level indicators of errors and updates.
  - Updated field value and/or error message for one or more fields.

From a high-level perspective, form data validation using OPA works as follows:

- At design time, the rulebase's data model is generated based on Form Definition metadata. The implementer saves the rulebase properties file locally and then uses it as a data model for the new rulebase.
- The newly-created rulebase is deployed on the Web Determinations Server.
- The rulebase is linked to the Form Definition:
  - A rulebase (Interview) is added to an Interview Set dedicated to form validation rules.
  - The new Validation Rule references the Interview.
  - The Validation Rule is linked to the Form Definition.



- At runtime, the integration layer invokes the rulebase via a web service using a generic WSDL. It is assumed that rulebase's data model remains unaltered by the rule writer and the rulebase is written following the assumptions described above.



Creating a Validation Rule Using OPA

Once the form structure is finalized and is not likely to be changed, the OPA business rule creation process may start. It contains a several steps:

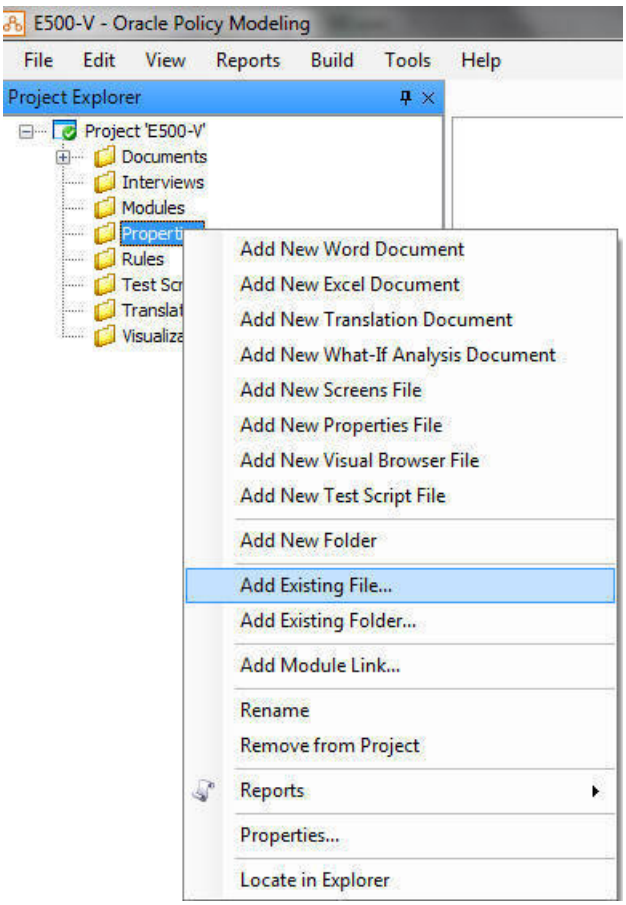
Step 1

- **Open the form in Form Designer and navigate to the Validation tab.**
- Click **Generate Rule Model**. This opens a Windows Explorer window.
- Locate the folder in which the model should be saved. Select "All" in the file type section.
- Provide a name for the rulebase model and save it with the extension .XSRC.

Step 2

Create a new project for the form validation in the Oracle Policy Modeling development environment.

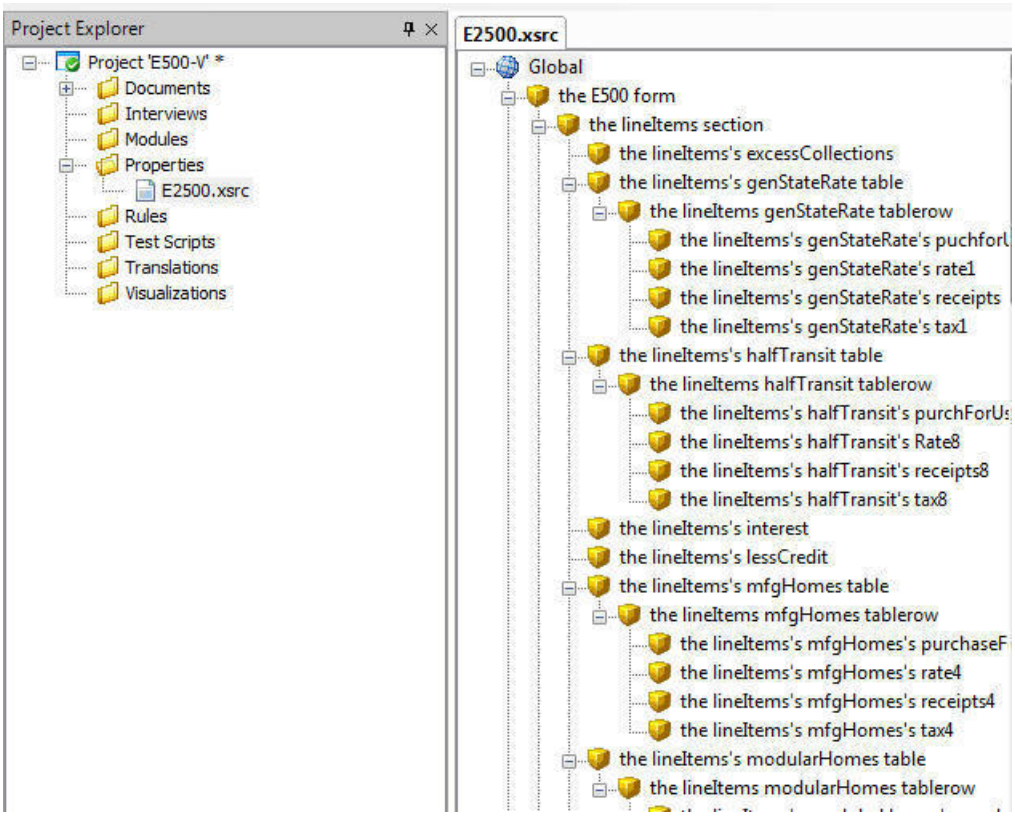
Incorporate the generated file as a property file, as shown in the following image:



Step 3

Open the property file.

Note that the data model has the same structure as the self service form definition. All form structure nodes are transformed into rule's entities. The entity name corresponds to the form node name and may be prefixed with the parent node name. This is done in order to ensure the uniqueness of the entity names within the generated model.



The following are used when the form definition is transformed into a rule data model:

- A ***Form*** node is converted into the first level of the hierarchy – Form entity. Form entity contains a set of attributes:
  - Some attributes, such as Action, Document Locator, Start Date, End Date are populated with the form data when form is passed to the rule for validation.
  - Some attributes, such as Form Contains Error, Form Error are expected to be populated by the rule and represent the outcome of the validation.
- A ***Section*** node is transformed into a ***Section entity*** . The Section entity contains the attributes relevant for the section.
  - *Section Contains Error* – This attribute is expected to be set by the rule to indicate that some elements of this section have failed the validation.
  - *Section Error Message* – This attribute is used to return a section-level error message.
  - *Section contains update* – This attribute is expected to be set by the rule to indicate that the determined values for some elements of this section were updated by the rule.
- Line node is translated into Line entity. Every line entity contain the attributes that can be used for validation/calculation:
  - A Line's submitted value attribute is used to pass to the rule the original value entered by the user.
  - A Line's determined value is set by the rule. It can contain the result of the calculations and/or validation, or it can be defaulted to the submitted value, depending on rule logic.
  - A Line's error message is used to pass back the error message for that line to the self service application.

You can modify the generated entity's text to make the entity more usable and intuitive. For example you may turn the generated "field_lineItems_firstPrepayment" into "the first prepayment". This simplifies the rule writing process and makes you rule much more readable. The following important limitations apply:

- Do not alter the generated Public Names in the data model. They are used by the integration between the self service application and the Oracle Policy Automation.

- The entity text must be unique within the data model.

Step 4

Set up the initial values for the fields that will be used during the validation.
To validate or pre-calculate the values of the form lines you may need to perform certain mathematical operations with the values of the fields. Note that OPA does not automatically set either blank values for text or zeroes for numeric fields. Instead, it defaults them all to the special value, "unknown". Since the formula cannot operate with that value, you may need to perform an explicit initialization that will take care of the form lines whose values are not provided.

If you use a line's determined value in your calculations, the initialization step will look like the following:

Table 2: Set Fields Determined Values

Total Gross Sales' Determined Value	
0	The total gross sales' submitted value is unknown.
The total gross sales' submitted value.	**Otherwise**

Step 5

If the form needs to report the error, add the section that checks if there were errors determined by the rule. If any errors are encountered, set the error indicator to *true*. The self service application uses this information to trigger the error message popup.

For example, your code may look like the following:

```
the form contains an error if
any
the taxpayer information contains an error or
the preparer information contains an error or
the line items contains an error or
the schedule a - deductions and exemptions contains an error or
for at least one of the instances of the schedule b
the schedule b - local tax computation contains an error

the taxpayer information contains an error if
the name's error message is known or
the account number's error message is known or
the address's error message is known or
the address 2's error message is known or
the city's error message is known or
the state's error message is known or
the country's error message is known or
the zip code's error message is known or
the filing start date's error message is known or
the filing end date's error message is known or
the ID number's error message is known or
the ID type's error message is known
```

Step 6

Process all fields that should be validated by your rule. The validation may verify that required value(s) were provided, or that the field has a proper value. If any errors are found, populate the attribute error message with the appropriate error text.

Example:

Table 3: Validate Taxpayer Information

The name's error message	
Error 1	The name's submitted value is unknown.

Uncertain	**Otherwise**

Result:

```
Error 1 = "<b>Taxpayer's name </b>must be provided"
```

Step 7

Perform the calculations.
The rule serves to validate the input and to perform pure calculations. The calculations are involved in both modes, but the difference may be illustrated by the following example:

```
Total = Total Gross Sales + Purchases Subject to Use Tax
the total's determined value = the total gross sales' determined value + the purchases
 subject to tax's determined value
```

If your rule validates that the total value entered by the user is correct according to this formula, your first step is to calculate the determined value and then to compare the submitted value and the calculation result and raise an error if they do not match.

The total's error message	
Error 15	The total's submitted value <> the total's determined value.
Uncertain	**Otherwise**

If your rule calculates the value, you need only set the determined values. The self service application updates the form data with newly-determined values and displays the updated form to the user when the validation is completed.

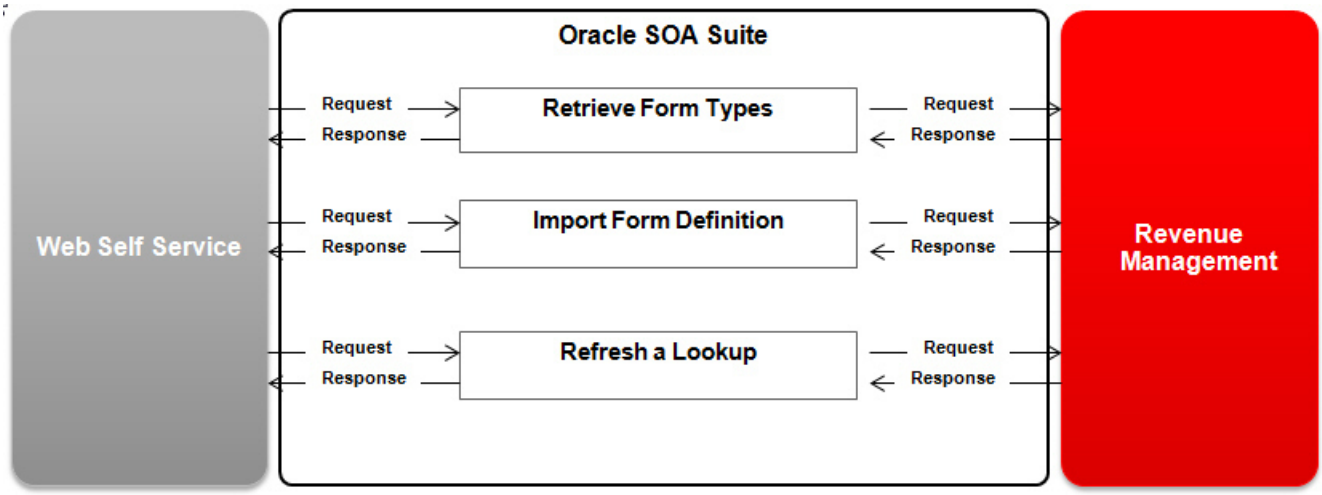
It is recommended to always populate a field's ***determined value*** with the calculation result.

Important Things to Keep in Mind

- You can add your own entities and attributes to the rule model. You can set up these attributes and use them in your rule. These attributes are only available within the scope of the actual rules; they are not passed back to the integration layer.
- Take into account that the self service user may trigger form validation multiple times while the form is only partially filled. You may wish to implement the "escape" logic that will skip certain validations and/or calculations if the input is not sufficient.
- Consider calculating the correct value right away rather than asking the taxpayer to provide the calculated input. The latter approach requires your rule to recalculate, compare the result with the input, and to return the error. The calculated lines can be defined as display-only on the Form Definition if their value is derivable based on the other lines. This will minimize fling time and improve the taxpayer experience.
- Always suggest a correct value for the line in the error message. You can use concatenation to do that. For example, you can use this syntax:
  - `Error 10 = The concatenation of " the total assessed tax amount" & the total tax's submitted value & " doesn’t mach calculated value " & the total tax's determined value & " . Total Tax = Tax + Local Tax."`
- If a determined value is provided when the rule returns the result, the self service application will display this updated value on the form. If you want to show the same value that was entered by the user, either do not populate line's determined value, or set it to the submitted value.

BPEL Processes

Integration Overview



The integration SOA composites invoke the Revenue Management System web services and send the response back to the self service portal application. The flows used for the form definition import are simple synchronous flows.

Implementing Form Design and Import

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

Web Services

In order to support the import of form definitions, the revenue management system is expected to implement the following web services:

Web Service	Description
TSRetrieveFormTypeDefinitions	This service retrieves definitions of a given form. The request contains a form type identifier and the user's language. The response contains: • Form structure (sections, lines, tables), with labels, data types and other details. • All the lookups used for this form type. • List of languages included in the import. Text-based information is expected to be retrieved in all languages supported by the revenue management system.
TSRefreshFormLookup	This service retrieves the up-to-date list values for a lookup. The request contains a lookup name and the response contains a list of values with descriptions in all languages supported by the revenue management system.

Web Service	Description
TSRetrieveActiveFormTypes	This service retrieves the list of form types available for import. The request includes the form category and the user's language. Each entry on the list is expected to contain: • **Form code** – Alphanumeric, no spaces, no special characters. • **Description.** • **Availability start and end date** (optional). Valid values for the form category are captured in the **FORMCAT** lookup.

Messages

The following messages are defined to support form design and import functionality:

Component	Messages

Message 51301 is used by the system for default confirmation text.

Configuration

Form Data Type

For each supported data type, specify the template field. These fields are used as reference fields on the form lines if the imported form definition does not include any information about the line's data type. Sample template fields are provided as follows:

Data Type	Field Name
BOOLEAN	DT_BOOLEAN
CURRENCY	DT_CURRENCY
DATE	DT_DATE
DATETIME	DT_DATETIME
LOOKUP	DT_LOOKUP
NUMBER	DT_NUMBER
TEXT	DT_TEXT

Messages

Message 51301 is used by the system for the default confirmation text.

System Options

Verify the following options:

DEF_FORM_CONFIRM_MSG defines the default message to be used for the form submission confirmation. The product is provided with the value 51301.

Email Definition

The self service product supplies the sample email definitions for tax and business registration form submission confirmations. Your implementation may wish to amend these or create new templates.

The email template includes text/HTML for email body. Use syntax `{message_details}` to mark a position at which the confirmation message details should be injected.

For additional details, see the email definitions topic in the *Self Service Settings and Configuration* section of this guide.

Form Process Control

Configure form process control records for all form categories supported by your implementation. The processing rules may be different for tax and registration forms.

- Indicate if form definition import functionality is supported by your implementation.
- Indicate if the revenue management system that maintains this category of forms supports on-demand validation of the form data.
- Select payment option to be used for the payment immediately following the form submission.
- Specify default confirmation email configured above.

Lookup

Your implementation may wish to create new lookups for the following reasons:

- For Forms Definitions imported from the revenue management system: You may decide to associate a limited list of values with a form line, even though the original form definition import did not include a lookup for this line. For example, a form line for filing type may have been imported as a regular text line. For the self service user's convenience, you may want to create a new lookup Filing Type and define list of valid Filing Type codes and descriptions, such as Single, Married Filing Jointly, etc.
- For Form Definitions defined in the self service product: Create a lookup for each form line that can be associated with the limited list of valid values.

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

System Options

Set up the following options:

DEF_CURRENCY - Lookup, references a main system currency.

DEF_COUNTRY_CODE - Lookup, references a country code for the country in which the revenue management authority operates. Used to display formatted address information.

UPL_FILE_TYPE - Free-form, comma-delimited list of file extensions. These file types can be uploaded by the self-service user.

OPA_VERSION - Free-form, specifies the OPA version currently in use for rulebase development. This value is used by the OPA Rulebase Model generator. **Example**: 10.4.2.18.

BPEL DVM Mapping

OTSS_FormCategory

An entry is provided for every Form Category included in the base product. The column ***OTSS_FormCategory*** contains the FORMCAT Lookup's values ***REGFORM*** and ***TAXFORM.***

For each entry, specify the corresponding translation values from your Revenue Management system in the column ***EXT_FormCategory.***

If your Revenue Management system does not use the form categories, leave the translation value blank. As a result, the <**formCategory**> node on the request xml would be empty.

If your implementation in the Revenue Management system is designed to use the same form categories as in the web self service portal application, delete the records from the DVM. As a result the value in the < **formCategory**> node on the request xml would be delivered 'as is'.

Examples:

The content of the node:

<formCategory>REGFORM**</formCategory>**

- With translation value XXX for REGFORM

<formCategory>XXX**</formCategory>**

- With blank translation value for REGFORM

<formCategory></formCategory>

- Without an entry in the DVM for REGFORM

<formCategory> REGFORM **</formCategory>**

OTSS_FormDataType

An entry is provided for every Data Type included in the base product. The column ***OTSS_FormDataType*** contains DATATYPE Lookup's values: ***BOOLEAN, TEXT, DATETIME, DATE, LOOKUP, CURRENCY, NUMBER*** and ***TIME***

For each entry, specify the corresponding translation values from your Revenue Management system in the column ***EXT_FormDataType.***

If your Revenue Management system does not use the data types, specify ***TEXT*** as translation value.

> **Note:** NOTE!!! It is mandatory to have data type node value translated as any attempt to import form definition without data types would result in error. TEXT is a default data type.

If your implementation in the Revenue Management system is designed to use the same data types as in the web self service portal application, delete the records from the DVM. As a result, the value in the <**dataType**> node on the request xml would be delivered 'as is'.

Examples:

The content of the node:

<dataType>BOOLEAN**</dataType>**

- With translation value XXX for BOOLEAN

<dataType>XXX**</dataType>**

- With blank translation value for BOOLEAN

<dataType></dataType>

- Without an entry in the DVM for BOOLEAN

<dataType>BOOLEAN **</dataType>**

OTSS_FormSectionOccurence

An entry is provided for every form occurrence type included in the base product. The column ***OTSS_Occurence*** contains OCCURENCE Lookup's values.

For each entry, specify the corresponding translation values from your Revenue Management system in the column ***EXT_FormSectionOccurence.***

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

Examples:

The content of the node:

<occurrence>SINGLE**</occurrence>**

- With translation value XXX for SINGLE

<occurrence>XXX**</occurrence>**

- With blank translation value for SINGLE

<occurrence></occurrence>

- Without an entry in the DVM for SINGLE

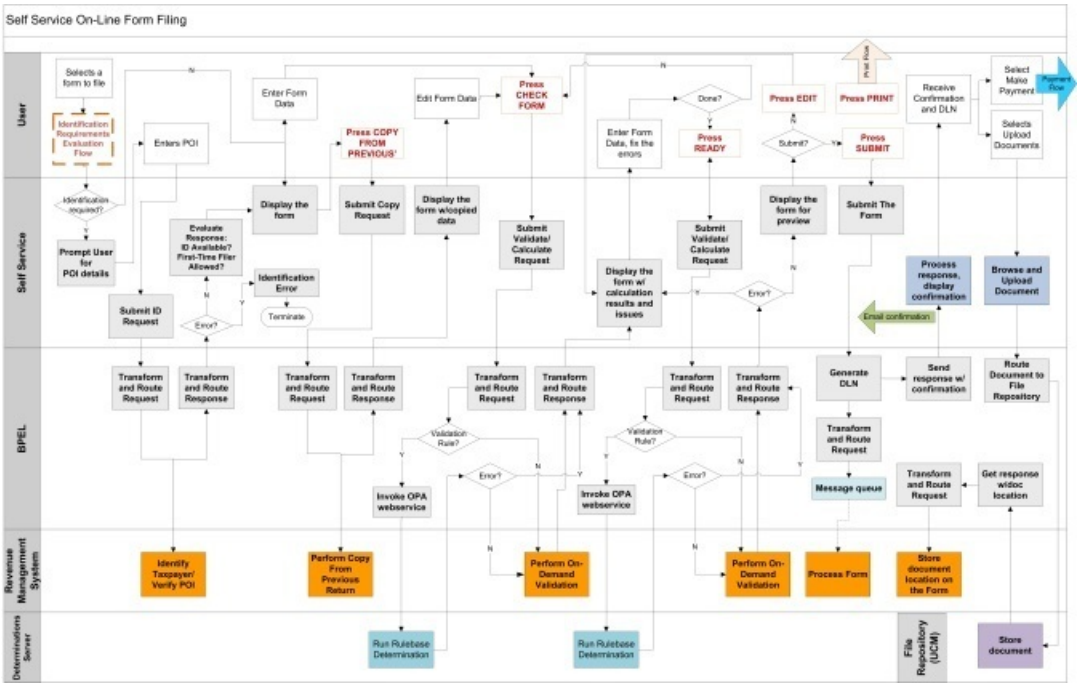
<occurrence>BOOLEAN **</occurrence>**

OTSS_MessageNumbers

Map the expected return message numbers from the Revenue Management system to the messages defined in the self-service portal application.

Form Processing

Filing the Form



The first step in the online form filing process is to select the form to file.

The Tax Forms and Registration Forms portal pages provide the user with a selection of *currently available Forms*.

When the user selects a form from the list, the system evaluates filing identification requirements (for details see the *Form Design* section) and the user may be prompted to *provide identification information* and/or select one of the tax accounts.

Upon successful identification, the form is rendered along with form instructions. The user may start entering the data and perform various *form actions*, including form data validation. Action buttons appear on the page according to the configuration. When the user clicks one of the action buttons, the *Process Tax Form* or *Process Business Registration Form* web service is invoked. The action code is populated on the request XML's <action> element.

The user may also *print the form data* before or after successful form submission.

Once the form is successfully filed, the user is prompted to make a payment (if required) and/or upload supporting documents according to the requirements. For additional details, see the *Form Design* section.

Available Forms

The portal includes a dedicated portal page for each major form category - tax forms and registration forms.

The page shows the list of available forms; the following is displayed for each form:

- **Form Description** is a hyperlink. Click initiates the online filing flow.
- **Form Advisor** is an icon. Displayed only if a form-level advisor is specified on the Form Definition.

Form availability is determined based on configuration (for details see the *Form Design* section).

Identification

The filing of a form may require different levels of self-service user identification:

- Available to all self-service users and not require any identification. For example, it could be a business registration form where the filing serves as a starting point in relationships between self-service user and the Revenue Management Agency.

In this case, the user selects the form to file and immediately begins the filing.

- Available to all self-service users, requires an identification, first-time filers not allowed. For example, it could be an application for an extension or any business-related form where the filing requires the existence of the record in the Revenue Management system.

In this case, the form always appears on the list.

If a **casual user** selects the form, the Identification Request linked to the form definition is invoked.

If a **registered user** selects the form, the subsequent flow depends upon whether this user also selected an account to work with. If no account has been selected, the user may either navigate to the Enrollment Summary page in order to pick an account or continue the filing. If the user chooses the latter option, he/she is further treated as a casual user and therefore the Identification Request linked to the form definition is invoked.

- Available to all self-service users who are first-time filers, identification required.

In this case, the form always appears on the list.

When a **casual user** selects the form, the Identification Request is invoked and the user is prompted to enter certain identification information. However, the back-end system is not expected to return any identifiers and may only check the validity of provided identification details - e.g. format, etc.

If a **registered user** selects the form, the subsequent flow depends upon whether this user also selected an account to work with. If no account has been selected, the user may either navigate to the Enrollment Summary page in order to pick an account or continue the filing. If the user chooses the latter option, he/she is further treated as casual user.

See the *Form Design* section for details on how to configure form identification requirements.

Form Actions

Copy From Previous Return (COPY)

The purpose of this action is to shorten and simplify the filing effort and pre-populate the form using the information stored in the back-end system.

Request XML contains self-service user identification details and/or access keys and form type. Optionally, it may contain filing period dates or other partial information.

Response XML contains a fully or partially filled form where the data is presumably copied from the previously filed form of the same kind. If no previous return was found, the response is expected to contain error/informational message

The availability of this action is based on the form definition.

Check Form Data (VALIDATE)

The purpose of this action is to verify the data entered by the user and perform as much calculations as possible.

During this action, the form is validated in multiple layers:

- The self-service portal application applies validation rules associated with form lines (for details see the *Form Design* section).
- The integration layer uses OPA-based Validation Engine and invokes OPA Rulebase associated with the form via web service.
- If the OPA-based validation is passed successfully, the BPEL process invokes the web service passing the form data to the back-end system.

Request XML contains self-service user identification details and/or access keys, form type and fully or partially populated form data. It also contains the name of the OPA Validation Rule to be invoked and the location of the OPA Server.

Response XML contains form data updated with the calculation/validation results and also a list of exceptions, if any.

The availability of this action is based on the form definition.

Ready (READY)

The purpose of this action is to run the form through the final round of validations and calculations. This action is available from the very beginning and the user can invoke it at any time. The process flow is similar to that of the action *VALIDATE*: the form is processed by OPA Validation Engine and then sent to the back-end system for further validation. There are two possible outcomes, depending on the web service response:

- No errors and/or exceptions -the form is considered ready for submission. No more changes are possible. The user can review the data and print it for his/her records.
- Errors/exceptions returned -the form is editable and may be further modified.

Request xml contains self-service user identification details and/or access keys and fully filled form data.

Response xml is expected to contain no errors and/or exceptions, only to echo back the form data.

This action is always available.

Submit (SUBMIT)

With this action, the form is finally delivered to the back-end system.

Request XML contains self-service user identification details and/or access keys and fully filled and validated form data.

The integration layer generates a **Confirmation ID** (identifier of the form submission transaction) and a **Document Locator Number** (identifier of the form itself that will be stamped on the form and used to track it across the enterprise).

The Document Locator Number is generated as follows:

- A sequence number is retrieved from the database, and a check sum is generated for the sequence number based on Luhn's algorithm and appended to the sequence number. The result is left-padded with zeroes and then a prefix is added.
- Sequence name, prefix and a final size of the document locator string are configurable and may be customized according to business requirements.
- The module responsible for Document Locator generation and validation can be customized or replaced the implementers (for additional information see *Service Configurations*.

The form delivery to the back-end system is either synchronous or asynchronous, depending on the configuration. In the latter case, the request is sent to the queue. For additional details see the *Form Design* section.

Response XML contains confirmation ID and message, Document Locator number and form data.

This action is always available.

Print Form

This button is associated with the special action and is available when the form is fully validated and ready to be submitted (after action *Ready*) or after final submission. In the latter case, the document locator number is displayed on the document.

The *Print Document* web service is invoked and the integration layer orchestrates the creation of a printable document that can be viewed and saved locally by the user. Out-of-the-box *printing of a form data* with Oracle BI Publisher is supported.

Custom Actions

Implementers may design additional form actions (for details see the *Form Design* section) and make them available during online filing. The processing associated with the custom action could be performed in the integration layer or in the back-end system or delegated to an external application via customized BPEL/SOA composite. The self service portal application invokes the web service with custom action populated.

Request XML contains self-service user identification details and/or access keys, form type and fully or partially populated form data. The <action> element value is set to the custom action's code. As with the **VALIDATE** action, the request contains the name of the OPA Validation Rule and the location of the OPA Server.

Response XML is expected to contain form data updated with the calculation/validation results and also a list of exceptions, if any.

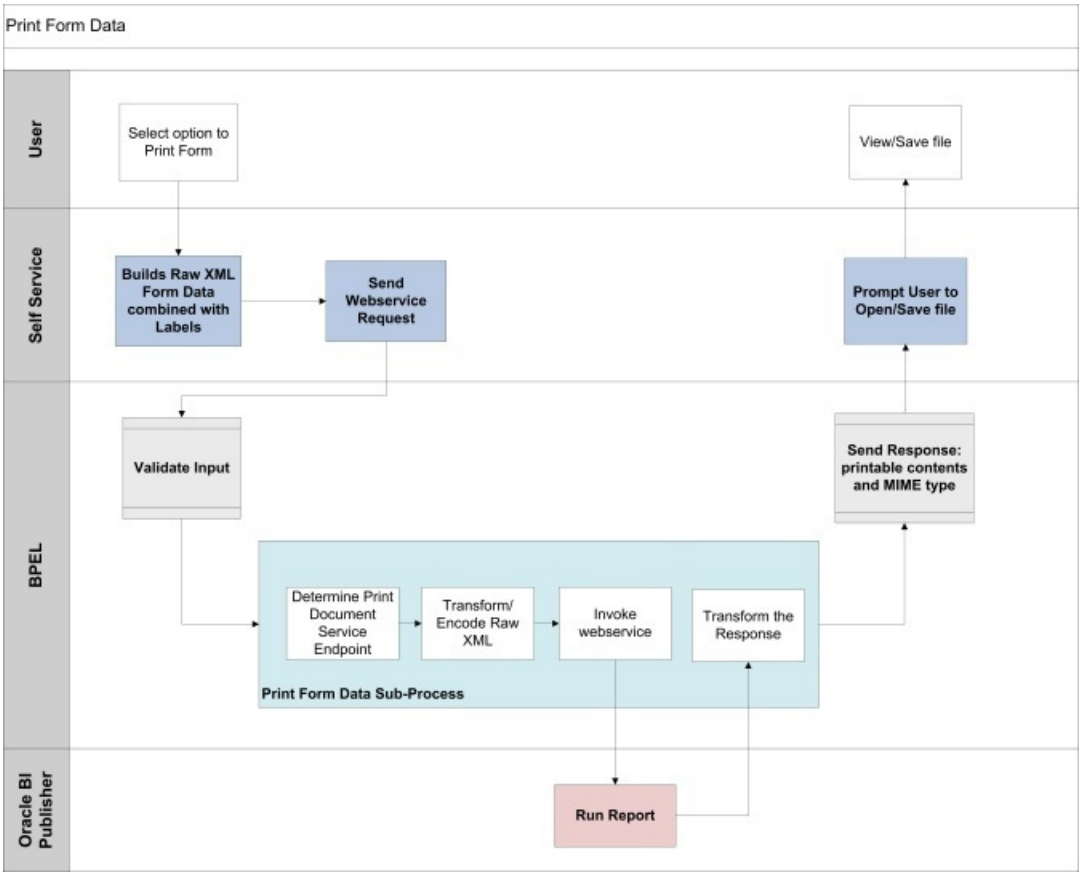
Custom action(s) are available based on the form definition.

Confirmation

Successful form submission results in the confirmation page displayed to the user. This page also provides the starting point for post-submission activities.

- Confirmation ID and details are is displayed; the contents of the confirmation depends on the form process submission method:
  - If the form was submitted synchronously, the page displays the confirmation details from the web service response.
  - If the form was submitted asynchronously, the system displays the confirmation message specified on the form definition.
- Confirmation email sent to the self service service user when the following conditions are satisfied:
  - Form definition indicates that email is required.
  - Form definition is referencing an email template.
  - The user provided an email address during the form submission.

Printing Form Data



When the *Print Document* web service is triggered from online Form UI, its **<documentData>** element contains form data and language-specific labels for form elements (generic structure of sections/lines/tables, raw valid XML, root node <documentData>).

```
<documentData>
<formType/>
<formTypeDescription/>
<formCategory/>
<currency/>
<customProperties> <!--list-->
<sequence/>
<propertyName/>
<propertyDescription/>
<value/>
</customProperties>
<documentLocator/ >
<documentLocatorLabel/>
<formData>
<id/>
<name >
<formDescription/>
<section> <!--list-->
<id/>
<name/>
<sectionLabel/>
<sequence/>
```

```
<line> <!--list-->
<field>
<id/>
<name/>
<lineLabel/>
<dataType/>
<value/>
</field>
</line>
<table> <!--list-->
<id/>
<name/>
<tableLabel/>
<tableRow> <!--list-->
<sequence/>
<field>
<id/>
<name/>
<fieldLabel/>
<dataType/>
<value/>
</field>
</tableRow>
</table>
</section>
</formData>
</documentData>
```

In the integration layer, the BPEL main flow invokes a sub-process that is responsible for document creation. Out-of-the-box solution includes a *sub-process that interacts with Oracle BI Publisher*.

The response from the sub-process is expected to contain Base64-encoded document content and mime type. The response is returned to the self-service portal and processed using ADF/browser native capabilities.

The BPEL process can be customized to work with different document generation software. See *Print Form Document* for additional details.

Document Creation with Oracle BI Publisher

The BI Publisher **ReportService** web service allows running reports using xml data supplied with the request.

The *BI Publisher report* **GenericFormDataModel** uses generic form data xml as its data model. A single *rtf template* provides basic layout for the form data and uses input values for section and field labels and for table titles and column headers.

The report displays form data "as is", i.e. the way it was entered online by the user.

Customization

The printing functionality is customizable.

Change Generic Layout

You can develop an alternative template using the same data model or add a new template to the report. You can also amend the BPEL process configuration and specify a new template file name in ***configuration.properties.***

Create Individual Reports for Forms

Generic form data can be transformed into form-specific XML using XSLT (see *example*). In this example, the generic form is transformed into an XML file with specific tags for sections and lines. The result of the transformation depends on form configuration.

You can *create new reports*, one per Form, using form-specific XML files as data models. Each report's layout design can be tailored for specific form data and/or mimic the official printable form design. Individual layouts can include additional regions and details. It may, for example, include a payment voucher or instructions, or any other form-specific content.

Decide on the layout naming convention so the layout name can be easily derived by the BPEL process. You can, for example, use the form type as the layout name.

In the BPEL process, modify the logic of the BI Publisher web service invocation. Instead of deriving the template name from the configuration, you can use <formType> as the layout name.

See *Create a Custom Report* in Appendix F for sample source, XML, and XSL files.

BI Publisher Report

The GenericFormDataModel report is provided with the product. This report's data model contains a single XML-based data set. The XML contains form data elements along with labels and descriptions, the document locator number, and the form type.

```
<documentData>
<formType/>
<formTypeDescription/>
<formCategory/>
<currency/>
<customProperties> <!--list-->
<sequence/>
<propertyName/>
<propertyDescription/>
<value/>
</customProperties>
<documentLocator/ >
<documentLocatorLabel/>
<formData>
<id/>
<name >
<formDescription/>
<section> <!--list-->
<id/>
<name/>
<sectionLabel/>
<sequence/>
<line> <!--list-->
<field>
<id/>
<name/>
<lineLabel/>
<dataType/>
<value/>
</field>
</line>
<table> <!--list-->
<id/>
<name/>
<tableLabel/>
<tableRow> <!--list-->
```

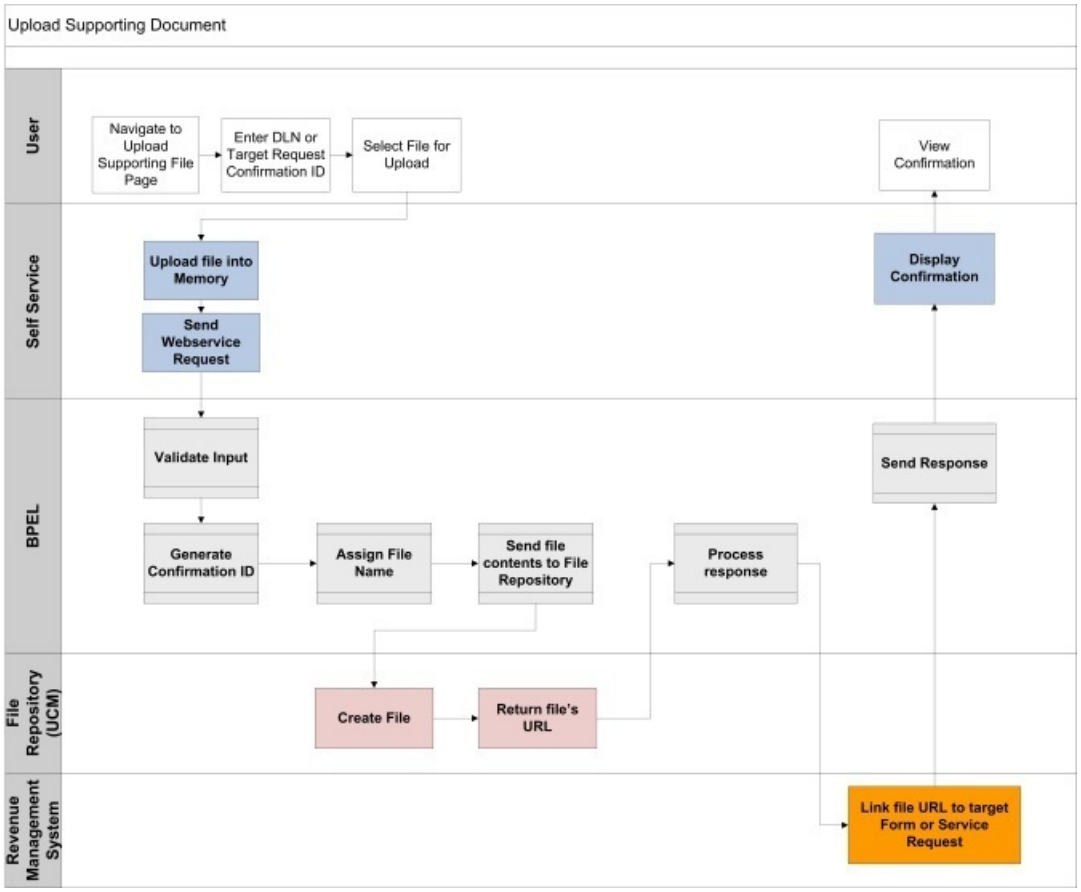
```
<sequence/>
<field>
<id/>
<name/>
<fieldLabel/>
<dataType/>
<value/>
</field>
</tableRow>
</table>
</section>
</formData>
</documentData>
```

The report supports all output formats permitted by BI Publisher.

An RTF template, GenericFormData.rtf, is provided with the product. It displays form line values and labels based on the information derived from the Data Model.

Note: The template does not conform with any actual "official" printable form layout.

Uploading Supporting Documents



The taxpayer may be asked to supplement a tax return or a business registration form with supporting documents.

The Self service application allows the user to initiate the document upload process in two ways:

- Upload supporting documents right after form submission. In this case, the upload can be done from the confirmation page. Upload instructions are displayed on the screen following form submission. The document locator number is taken from the *Process Form* web service response.
- Upload supporting documents later, from a dedicated portal page. On this page, the user is asked to select the Form Type (in order to display upload instructions) and also provide the Document Locator Number. This page provides an alternative option to upload a file using a confirmation ID or a previously-submitted request.

The user is prompted to browse the local directories and select a file for upload. The upload action can be repeated multiple times, but only one file can be uploaded at a time.

Permissible file types are defined in the System Configurations Option **UPL_FILE_TYPE**. The maximum size of the uploaded file is defined in the portal application properties XML. The initial limit is set to 10MB.

The Upload Document web service is invoked and the contents of the file are sent to the Integration layer, base64-encoded. Either the document locator number or a "target" request confirmation ID is included in the request.

The file name is assigned based on the target object's identifier so it can be cross-referenced and identified in the repository.

The back-end system may establish the link between the form and an uploaded document using the Document Locator Number, upload Confirmation ID, and file location.

The response from the back-end system is expected to contain confirmation details.

Document Repository

The provided solution uses the Web Content Manager Server as a document repository.

The supporting documents uploaded by taxpayers ***should not*** be placed in the same Web Content Manager Server repository that holds public documents, such as web site content. Taxpayer documents should be uploaded to a separate, secured repository (for example, a different UCM server). These documents will have different security settings and will not likely have the sample properties listed above.

Making a Payment

After a form is successfully submitted, there may be an option to make a payment, depending on the form type's configuration.

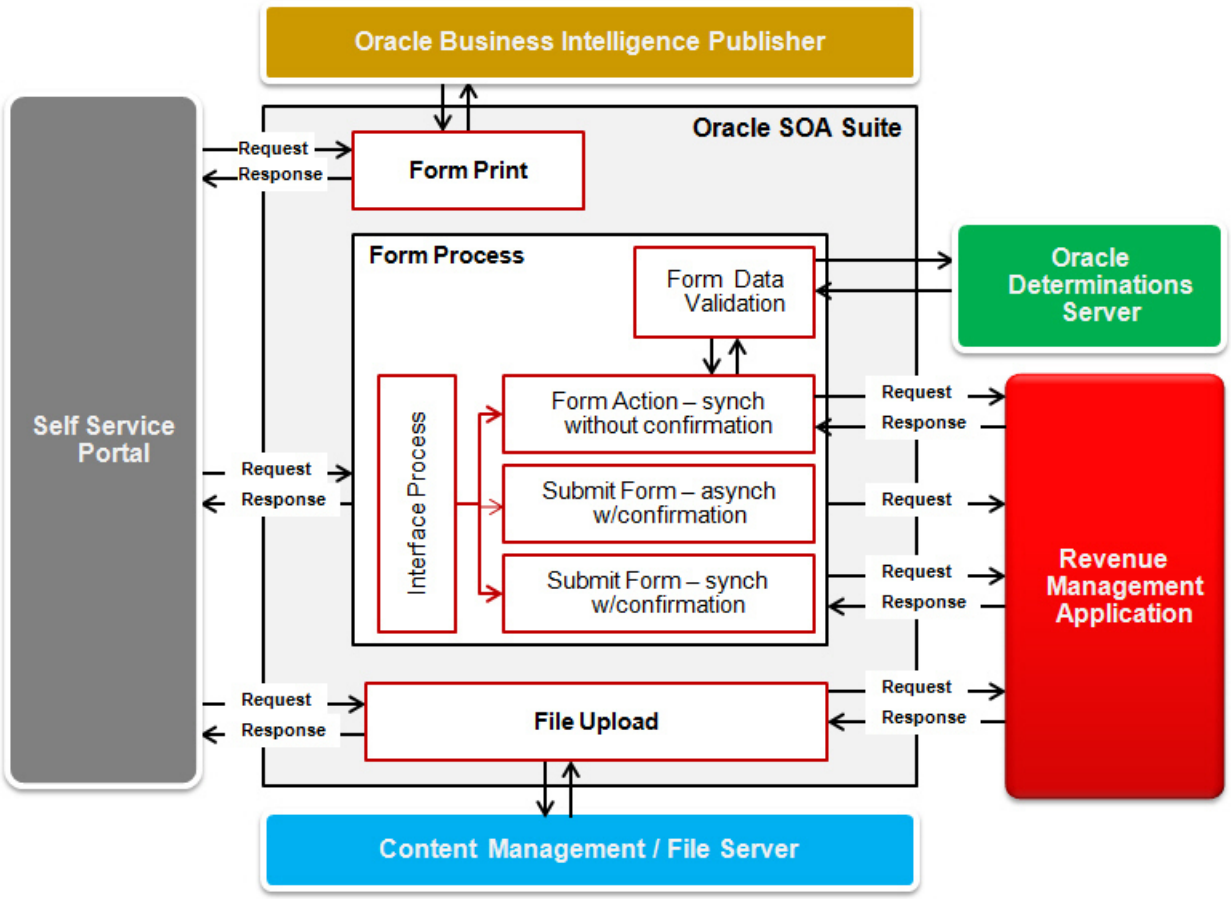
A proposed payment amount is determined according to *Form Definition*. The amount can be either a pre-defined fee or a calculated amount derived from the form data.

From the confirmation page, a user can select the option to pay and trigger the one-time payment flow. On the payment details page, the Document Locator Number is pre-populated from the *Process Form* web service response.

At the time of payment, the actual form may not yet be posted (*asynch process*). In this case, additional reconciliation logic should be triggered in the back-end system to link the payment to the form once the form is posted.

BPEL Processes

Integration Overview



The integration SOA composites invoke the revenue management system web services and send the response back to the self service portal application. Form processing flows include Document Locator Number generation.

The flow combines synchronous and asynchronous patterns.

Implementing Form Processing

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

Web Services

To support online form processing, the revenue management system is expected to implement the following web services:

Web Service	Description
TSProcessTaxForm	This service supports the process of online filing. The request contains an XML fragment representing the form data: `<formData>`

Web Service	Description
	<id/> <name/> <docLocatorNumber/> <section> (list) <id/> <name/> <sequence/> <line> (list) <field> <id/> <name/> <dataType/> #list <value/> </field> </line> <table> (list) <id/> <name/> <tableRow> (list) <sequence/> <field> (list) <id/> <name/> <dataType/> <encrypt/> <value/> </field> </tableRow> </table> </section> </formData> The process flow is dependent on the following actions: • Actions **COPY**: The request is processed synchronously. The expected response from the revenue management system includes an updated form data fragment. • Actions **VALIDATE**, **READY**, and **custom actions**: The request is processed synchronously. The integration invokes a sub-flow that interacts with Oracle Web Determinations and invokes an OPA rulebase via a web service. The expected response contains an updated form data fragment and/or list of form data exceptions. • Action **SUBMIT**: The request is processed asynchronously or synchronously, depending on the form configuration. The form is either sent to the revenue management system or queued for later processing. The expected response contains confirmation details and the document locator number.
TSProcessRegistrationForm	This service is an exact clone of the TSProcessTaxForm service. It provides: • **Better performance**. The workload is split between two services in large-scale implementations supporting both taxes and business registration. • **Implementation convenience**. The business registration flow in the integration layer and/or in the revenue management system can be customized on its own.

Support for creation of a printable file is also provided by the product. See *Print Form Data* for more details on configuration requirements.

Web Service	Description
TSPrintDocument	This service is invoked from the self service portal application and is handled in the integration layer. The SOA Composite interacts with the document generation system and sends back base64-encoded document contents. The request contains the source document data in a form of raw XML. The response includes printable document contents (base64) and the document's MIME type.

Web Service	Description

	When invoked with the purpose of printing the form, the source document data is populated with form details, including labels and descriptions:
	
	`<formType/>`
	`<formCategory/>`
	`<currency/>`
	`<formTypeDescription/>`
	`<customProperties> (list)`
	`<sequence/>`
	`<propertyName/>`
	`<propertyDescription/>`
	`<value/>`
	`</customProperties>`
	`<documentLocator/>`
	`<documentLocatorLabel/>`
	`<formData>`
	`<id/>`
	`<name/>`
	`<section> (list)`
	`<id/>`
	`<name/>`
	`<sectionLabel/>`
	`<sequence/>`
	`<line> (list)`
	`<field type="group">`
	`<id/>`
	`<name/>`
	`<lineLabel/>`
	`<dataType/>`
	`<value/>`
	`</field>`
	`</line>`
	`<table> (list)`
	`<id/>`
	`<name/>`
	`<tableLabel/>`
	`<tableRow> (list)`
	`<sequence/>`
	`<field type> (list)`
	`<id/>`
	`<name/>`
	`<fieldLabel/>`
	`<dataType/>`
	`<value/>`
	`</field>`
	`</tableRow>`
	`</table>`
	`</section>`
	`</formData>`

Messages

The following messages are defined to support form definition functionality:

Component	Messages
Form Import	10003, 21011, 21012, 21013, 21014, 21015
PROCESS_TAX_FORM	10003, 22010, 22011, 22012, 23011
PROCESS_REG_FORM	10003, 22010, 22011, 22012, 23011
UPLOAD_DOCUMENT	23003, 23004, 23006, 23011

Configuration

In order to enable the supported service requests you may add/modify the following configurations:

System Options

Set up the following options:

- **DEF_CURRENCY** - lookup; references a default system currency.
- **DEF_COUNTRY_CODE** - lookup; references a country code in which the revenue management authority operates. Used to display formatted address information.
- **UPL_FILE_TYPE** - free-form, comma-delimited list of file extensions. These file types can be uploaded by the self-service user.
- **OPA_VERSION** - free-form; the OPA version currently in use for rulebase development. This value is used by the OPA Rulebase Model generator.

Advanced Navigation

In order to embed link to a specific form within HTML content, use the following syntax:

```
/faces/service/form?FRM=<Form Type Code>
```

BPEL DVM Mapping

OTSS_Currency

An entry is provided for every Currency Code used in the base product. The column ***OTSS_CurrencyCode*** contains Currency Lookup's values.

For each entry, specify the corresponding translation values from your Revenue Management system in the column ***EXT_CurrencyCode*** .

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

Examples:

The content of the node:

<currency>US**</currency>**

- With translation value XXX for US
- <**currency**>XXX</**currency**>
- With blank translation value for US

<currency></currency>

- Without an entry in the DVM for US

```
<currency>US</currency>
```

OTSS_MessageNumbers

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

Chapter 10

Track Your Transaction

Overview

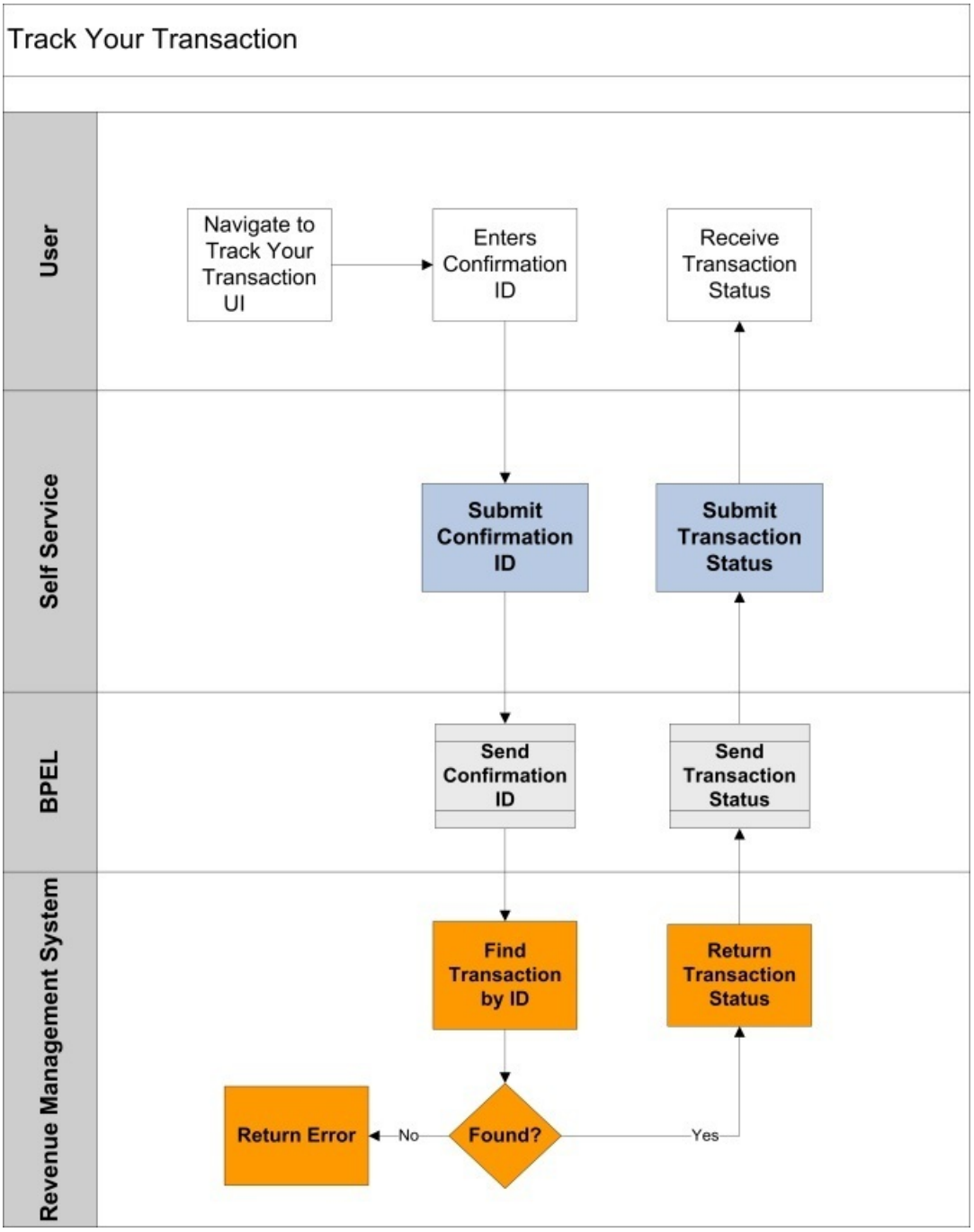
When a self-service user performs an operation that requires some action in the revenue management system he or she receives an acknowledgment that the request went through successfully. This acknowledgement is called a "confirmation". It contains a unique ID and a message with the current state of the 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 (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 11

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 taxpayer'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 Oracle Policy Administration (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 Interactive Tax Assistant. 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.

The interviews may be translated to one or more languages. Portal application supports launching the tax assistance with the language (locale) currently selected by the self-service user. If no translation is available, the default locale is used.

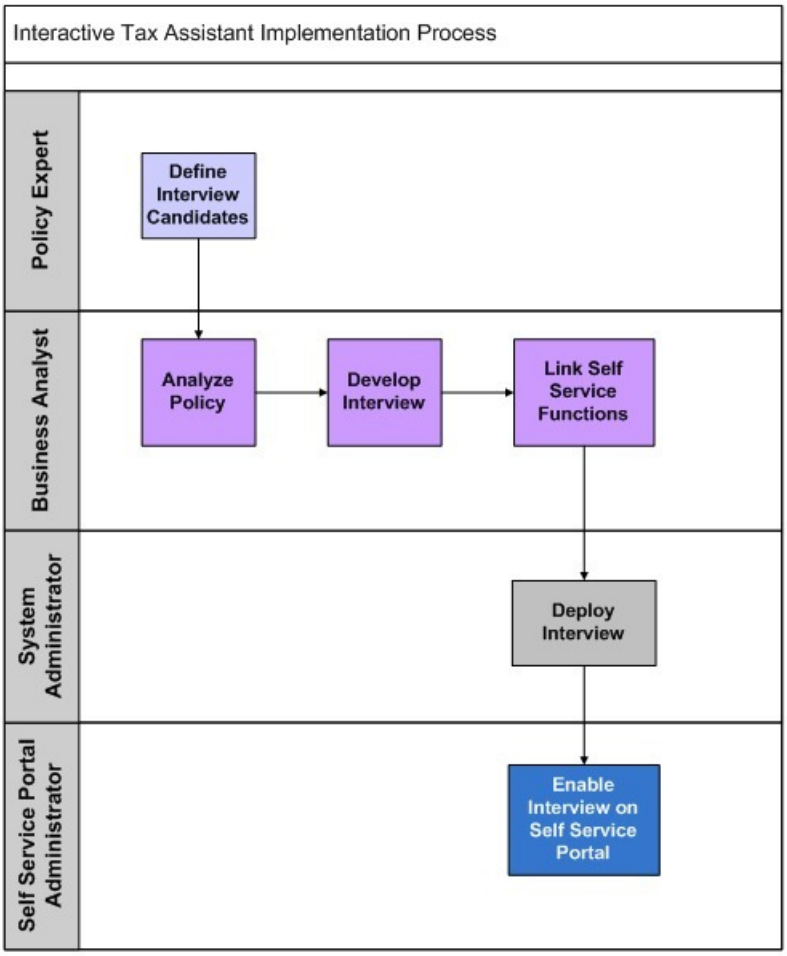
After web interviews are developed, tested, and deployed on a production server, they become available to taxpayers on the ITA page.

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. The implementers 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.

Interview Set Category indicates the intended purpose of the interviews in this set.

- **Tax Assistance** sets belonging to this category contain Interactive Web Interviews exposed on Tax Assistance portal page and throughout the system.
- **Filing Advisor** sets include Interactive Web Interviews used by form filing feature. These interviews are launched from the various regions of the online form.
- **Data Validation** sets include Web Determinations that may be associated with Validation Rules and invoked via web service by Form Validation Engine.

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. Use the following syntax:

- For sets containing interactive interviews, use `[host]:[port]/web-determinations`
- For sets containing data validation rulebases, use `[host]:[port]/determinations-server`

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.

Active indicates that the interview is accessible online in the current user's language. The interview should be activated separately for each deployed translations.

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 12

Oracle Policy Automation

Oracle Policy Automation Overview

The Interactive Tax Assistant, Form Filing Advisors and Form Validation engine are implemented using the Oracle Policy Automation (OPA) solution, converting the policies into a series of a problem-solving dialogs.

Oracle Policy Automation comprises five main components:

- Oracle Determinations Engine
- Oracle Determinations Server
- Oracle Web Determinations
- Oracle Interview Portlet
- Oracle Web Determinations Interview Engine

For additional details on Oracle Policy Automation features and functionality, visit the product documentation at *http://www.oracle.com/technetwork/apps-tech/policy-automation/documentation/index.html*.

Chapter 13

SOA/BPEL Integration

Integration Overview

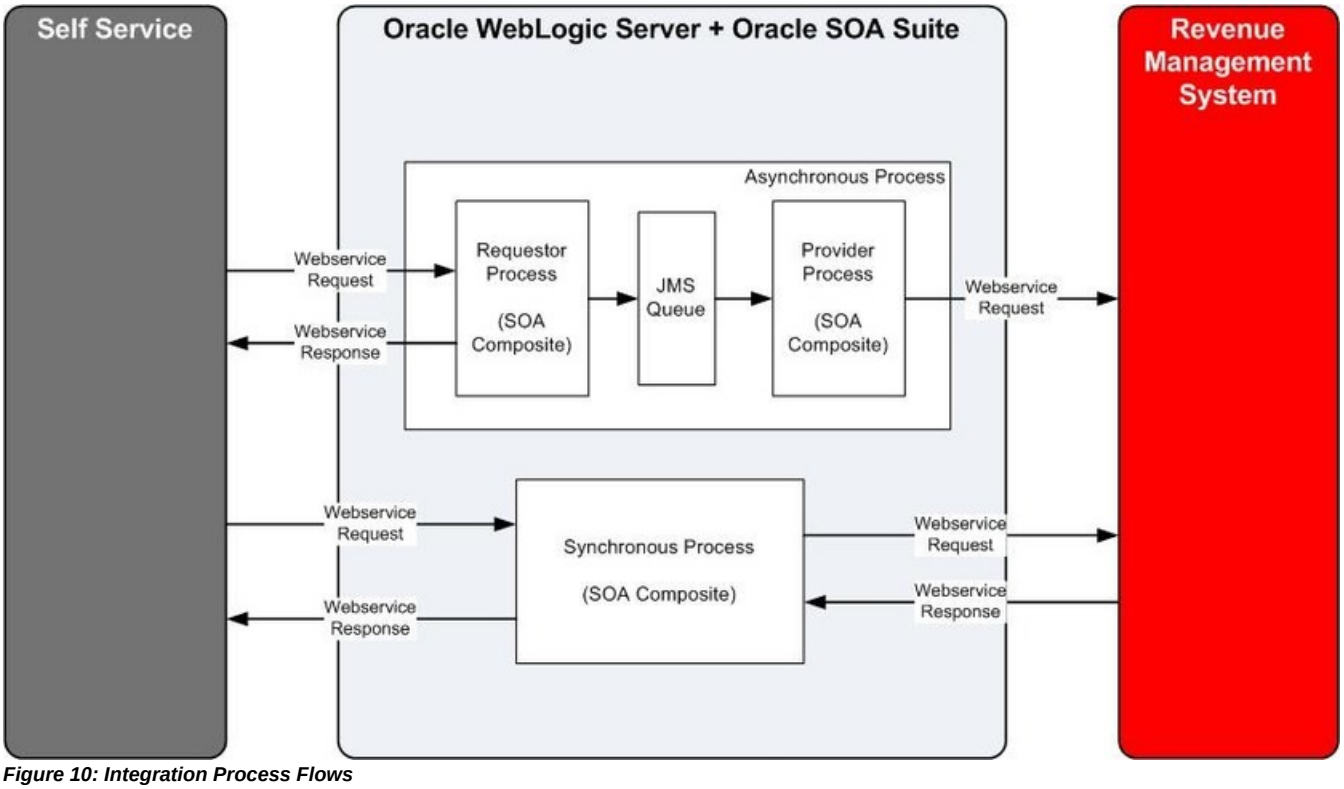


Figure 10: Integration Process Flows

This integration is an AIA Direct Integration.

All interaction between the self-service application, the integration layer, the revenue management system, and other solution components 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 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:

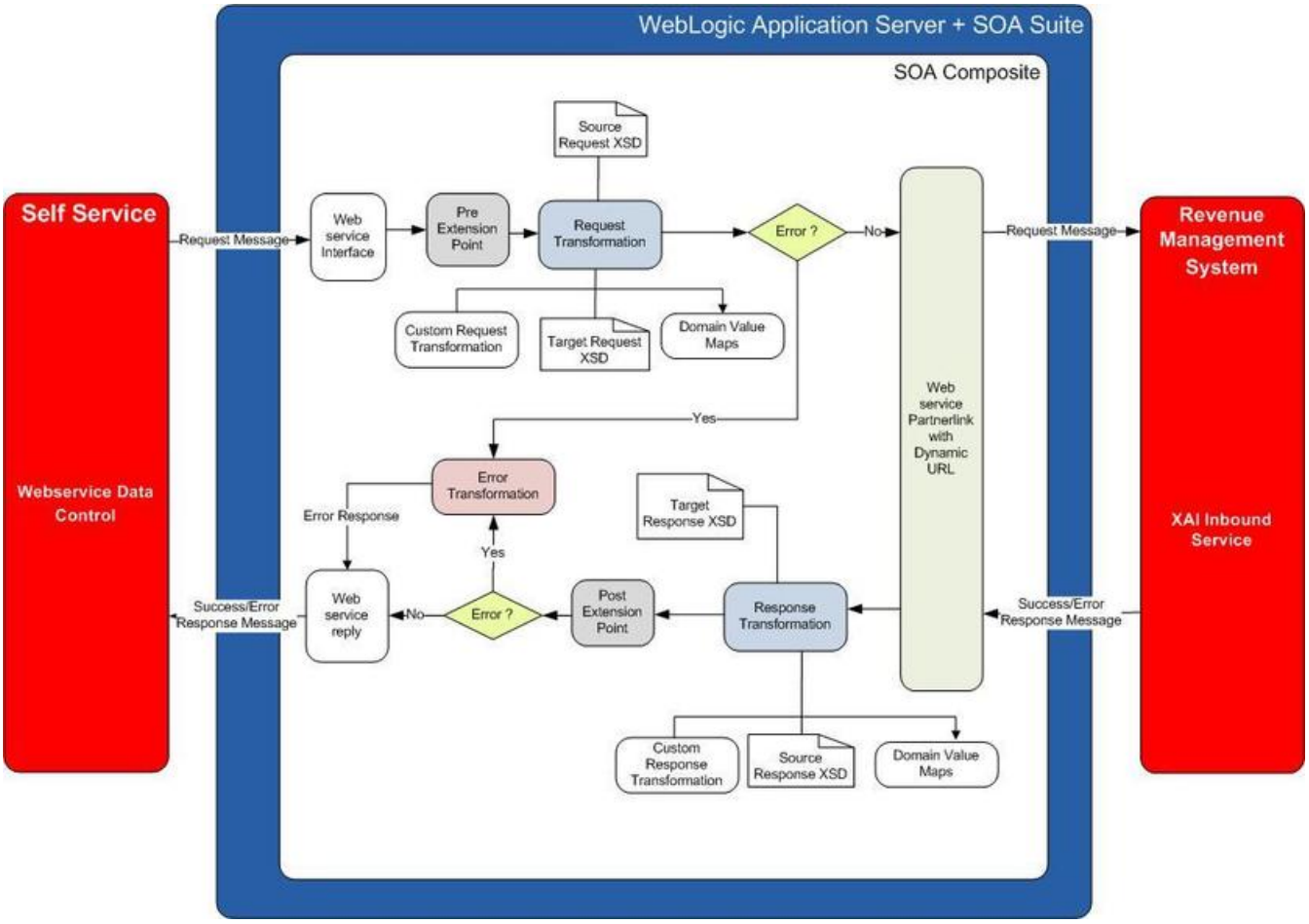
- Synchronous flow without confirmation ID.
- Synchronous flow with confirmation ID.
- Asynchronous flow with confirmation ID.

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

User enrollment implementation includes special synchronous flows with enrollment ID generation, enrollment data updates, and interaction with Oracle BPM Worklist.

The integration for online form processing features document locator number generation, interaction with Oracle Determinations Server, Content Management Server, and Oracle Business Intelligence Publisher, and includes special synchronous flows.

Synchronous Flow Without Confirmation ID

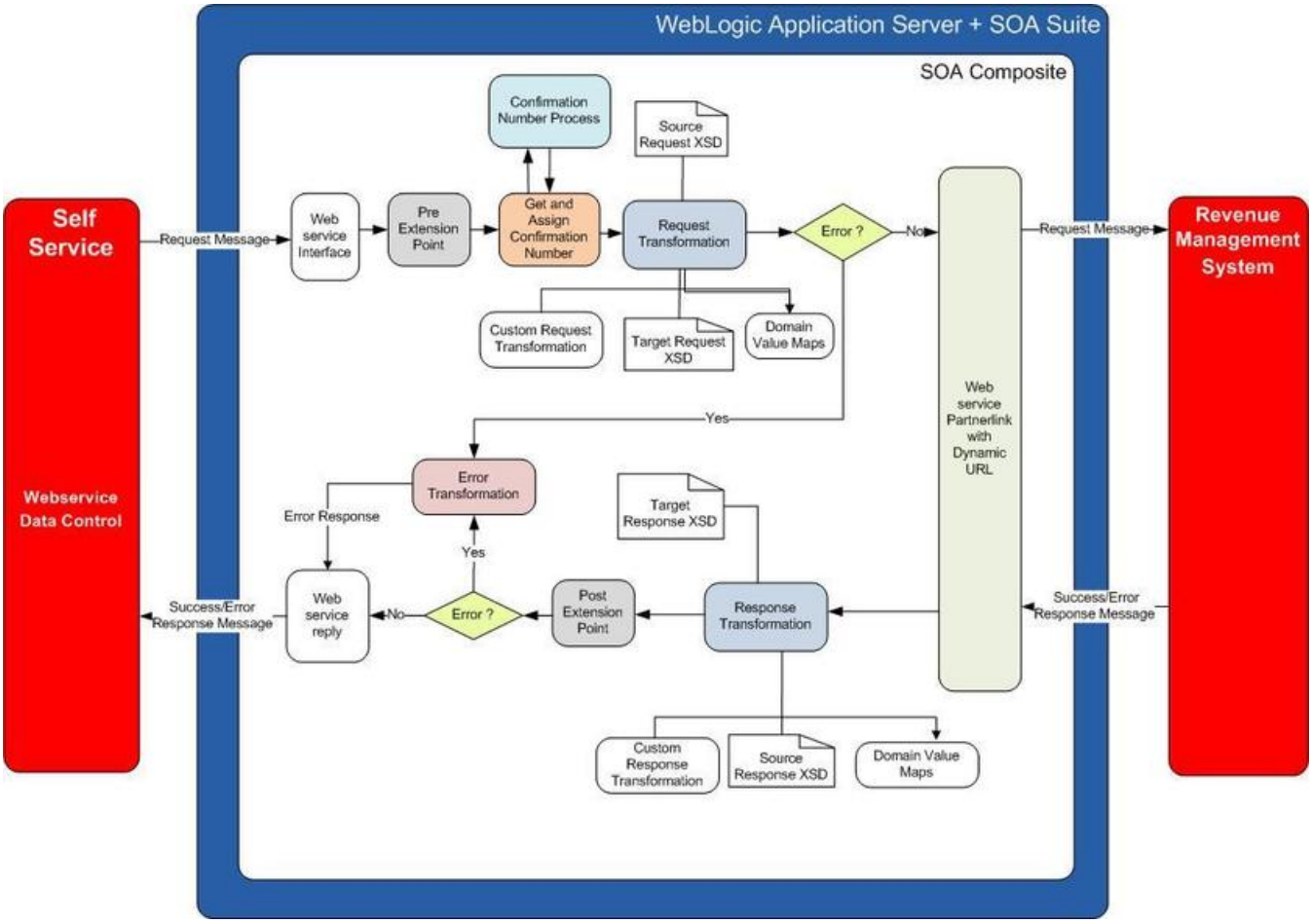


- The self-service application initiates the flow and sends a request (web service) to the integration layer.
- The integration SOA composite gets the XML message and transforms it from the self-service system format into the target system format.
- The SOA composite then forwards the request (web service) 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

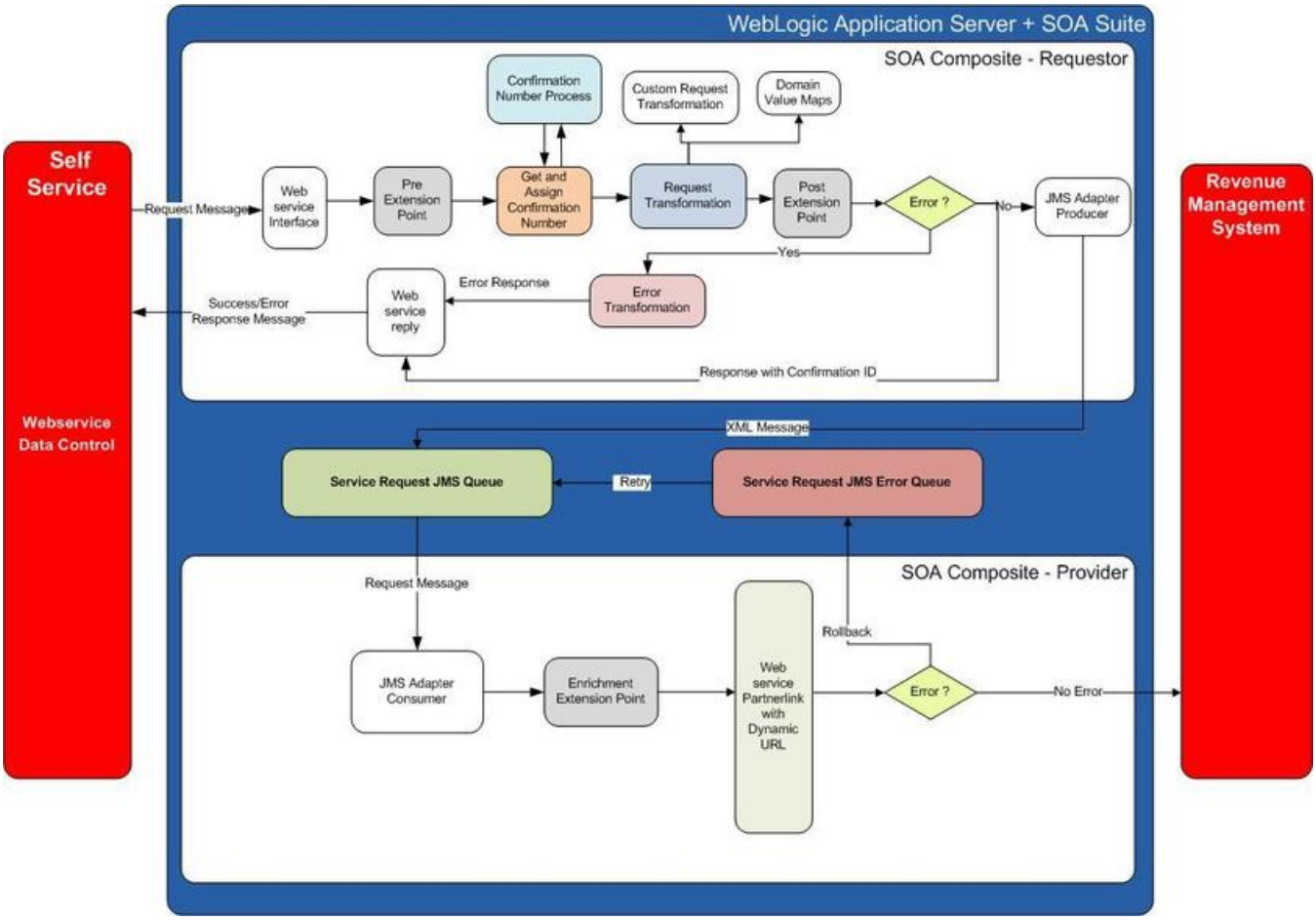
This flow includes an additional step during response transformation; it stores a record in the internal staging table. Otherwise, this pattern is the same as *Synchronous Flow Without Confirmation ID*.

Synchronous Flow With Confirmation ID



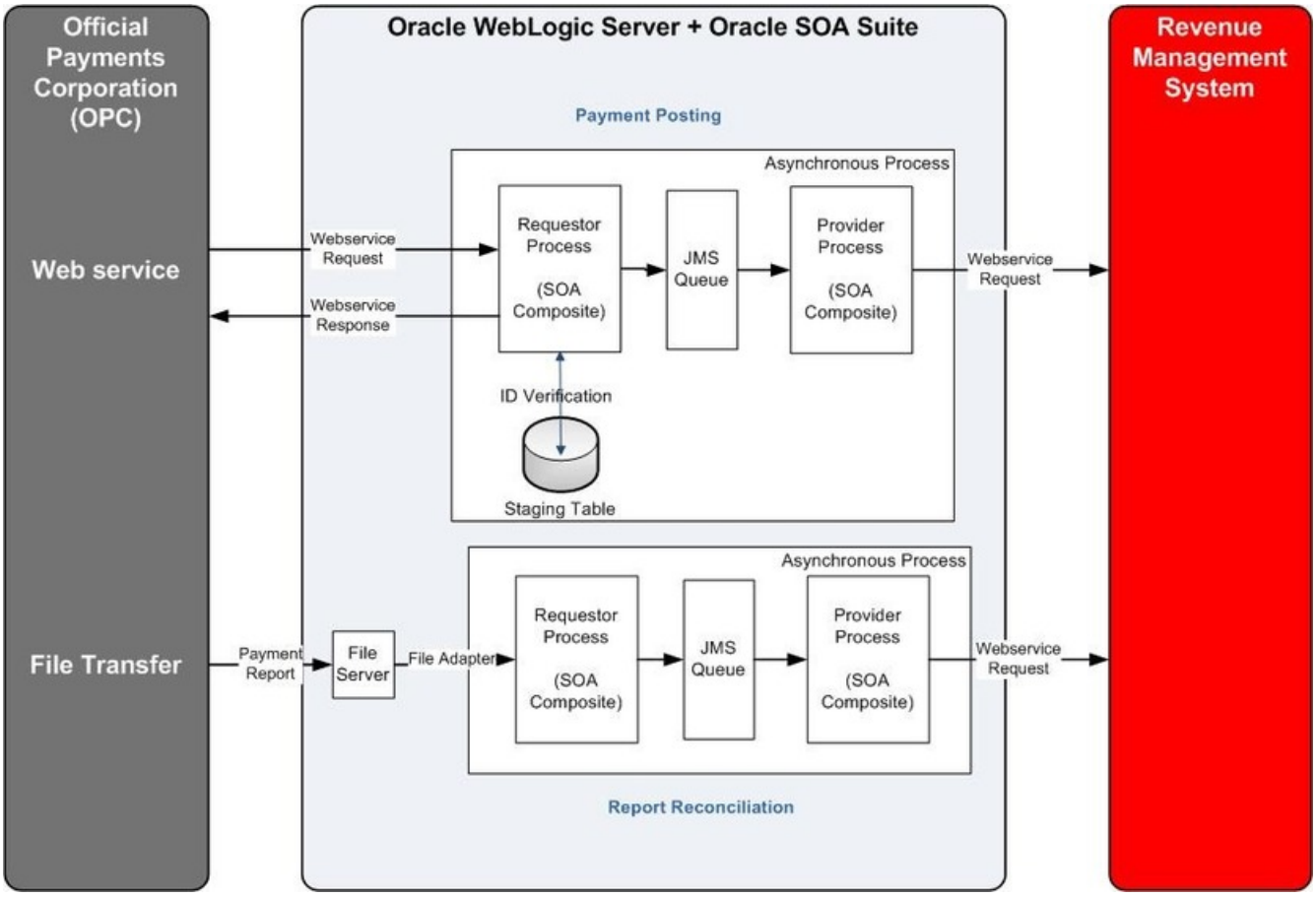
- The self-service application initiates the flow and sends a request (web service) to the integration layer.
- The integration SOA composite gets the XML message and transforms it from the self-service system format into target system format.
- The *Confirmation Number Utility Service* is invoked and 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.

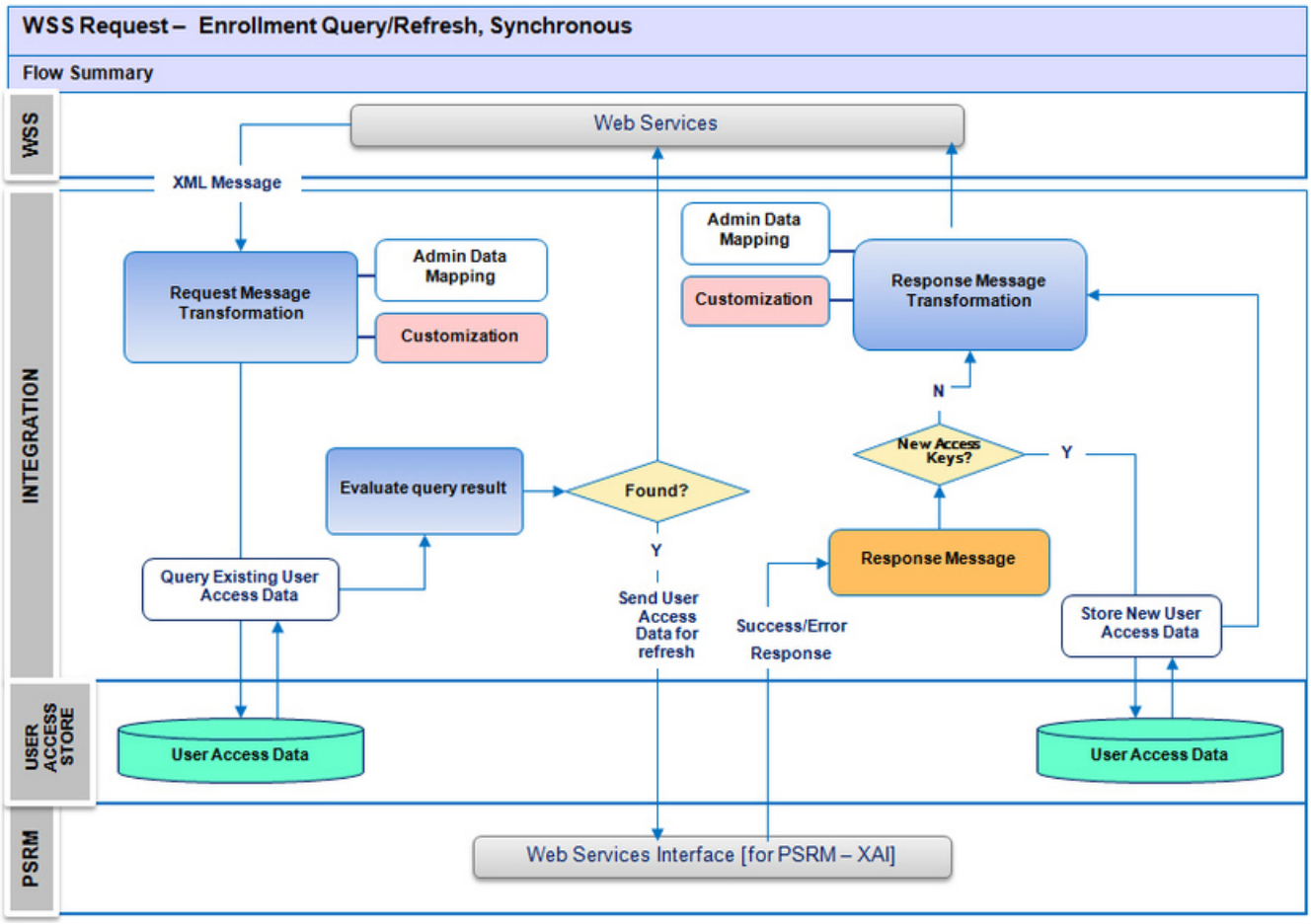
The Payment Posting process includes:

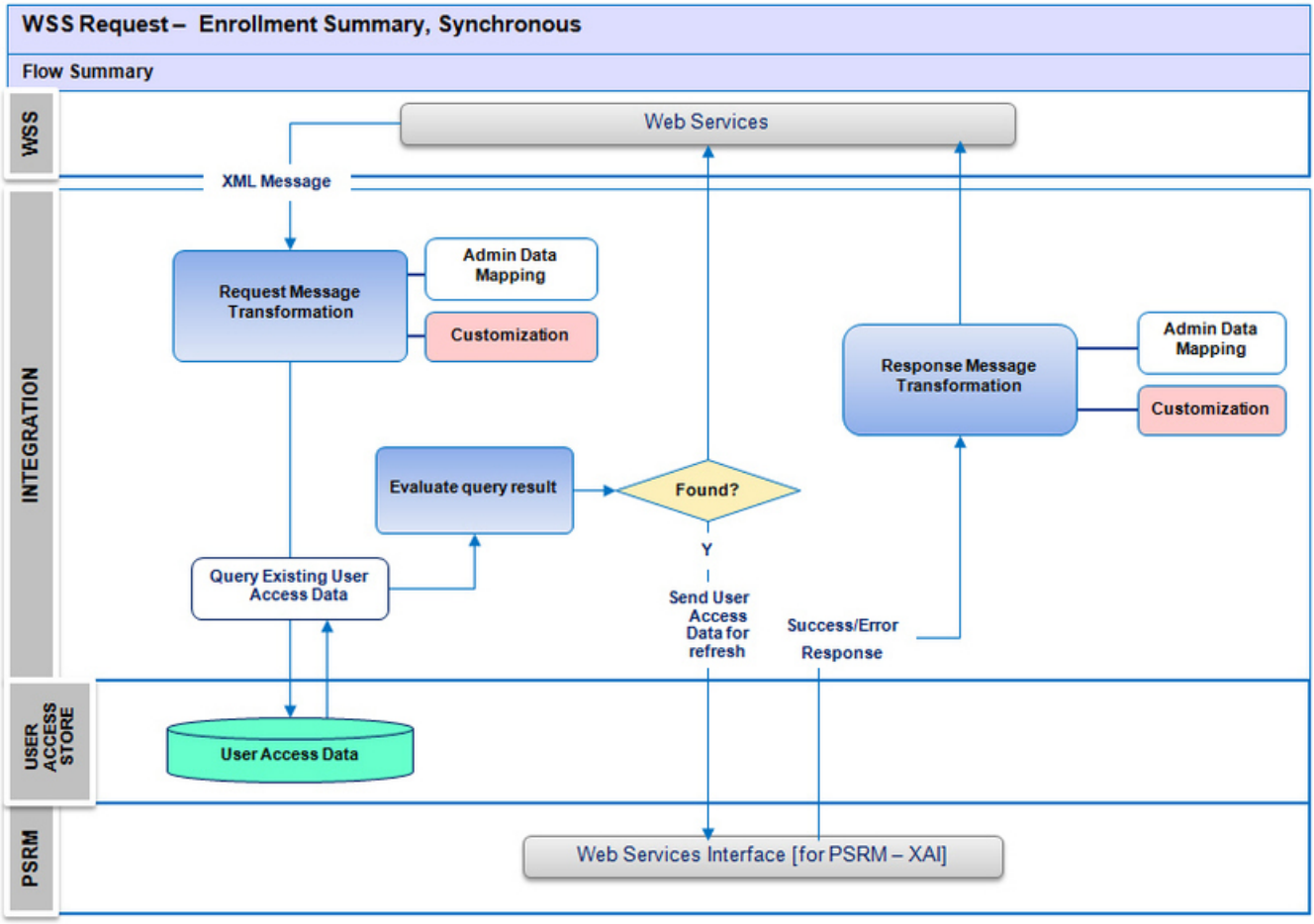
- Transaction verification against the record stored in the staging table.
- The special logic for confirmation number creation.

Enrollment Request Flows

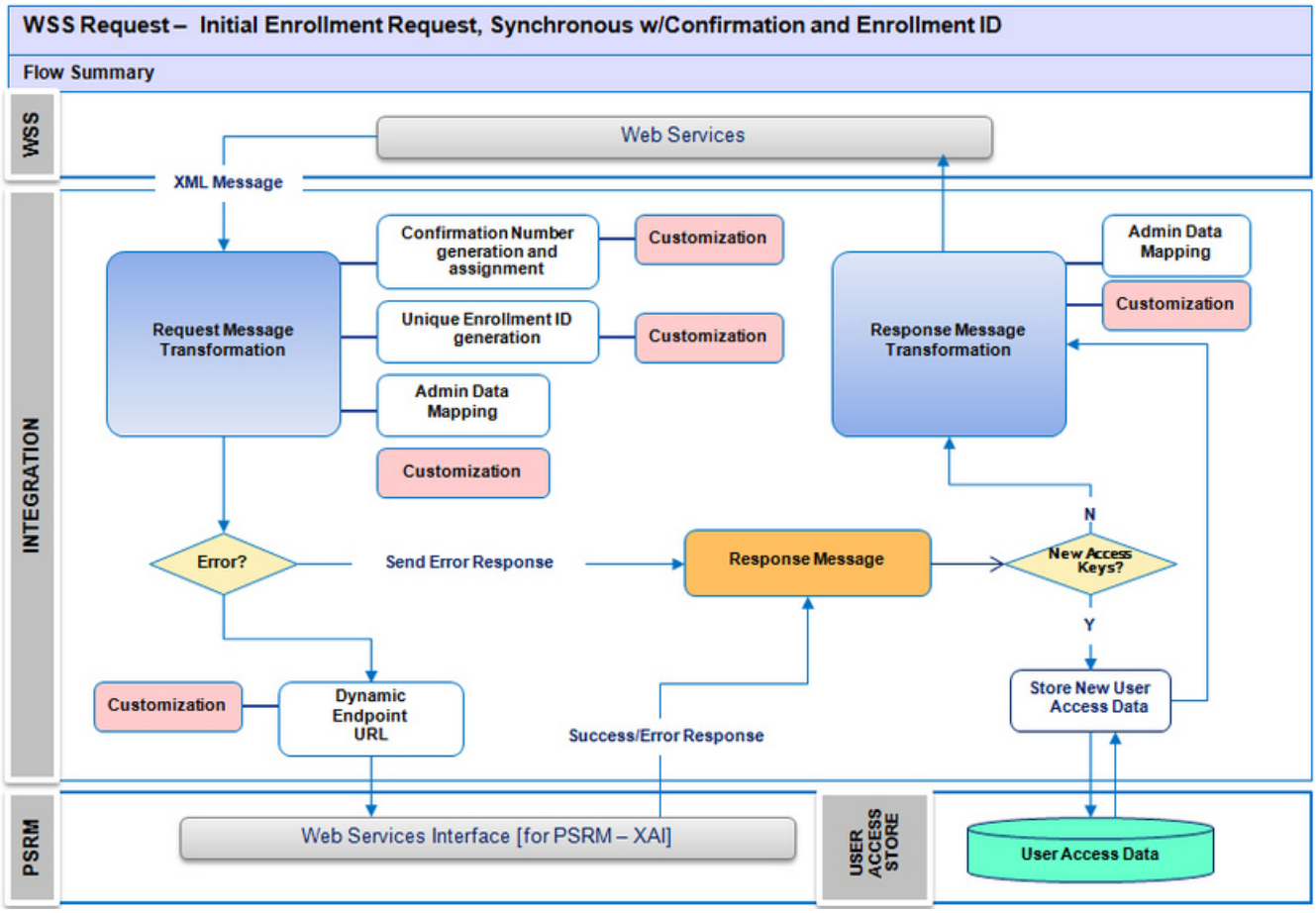
This integration contains three special flows that include interaction with the User Access Store and Oracle BPM Worklist.

Query/Refresh Enrollment and **Retrieve Enrollment Summary** flows are synchronous; both flows invoke a special utility to query the user access data. The refresh enrollment summary also invokes this utility for updates.


WSS Request – Enrollment Query/Refresh, Synchronous
Flow Summary
WSS
Web Services
XML Message
INTEGRATION
Admin Data Mapping
Request Message Transformation
Customization
Admin Data Mapping
Customization
Response Message Transformation
N
New Access Keys?
Y
Evaluate query result
Found?
Y
Response Message
Query Existing User Access Data
Send User Access Data for refresh
Success/Error Response
Store New User Access Data
USER ACCESS STORE
User Access Data
User Access Data
PSRM
Web Services Interface [for PSRM – XAI]



The **Initial Enrollment Request** process includes confirmation ID and Enrollment ID generation logic. The process invokes a sub-flow to perform enrollment errors and count verification and also invokes the special utility to update the User Access store.

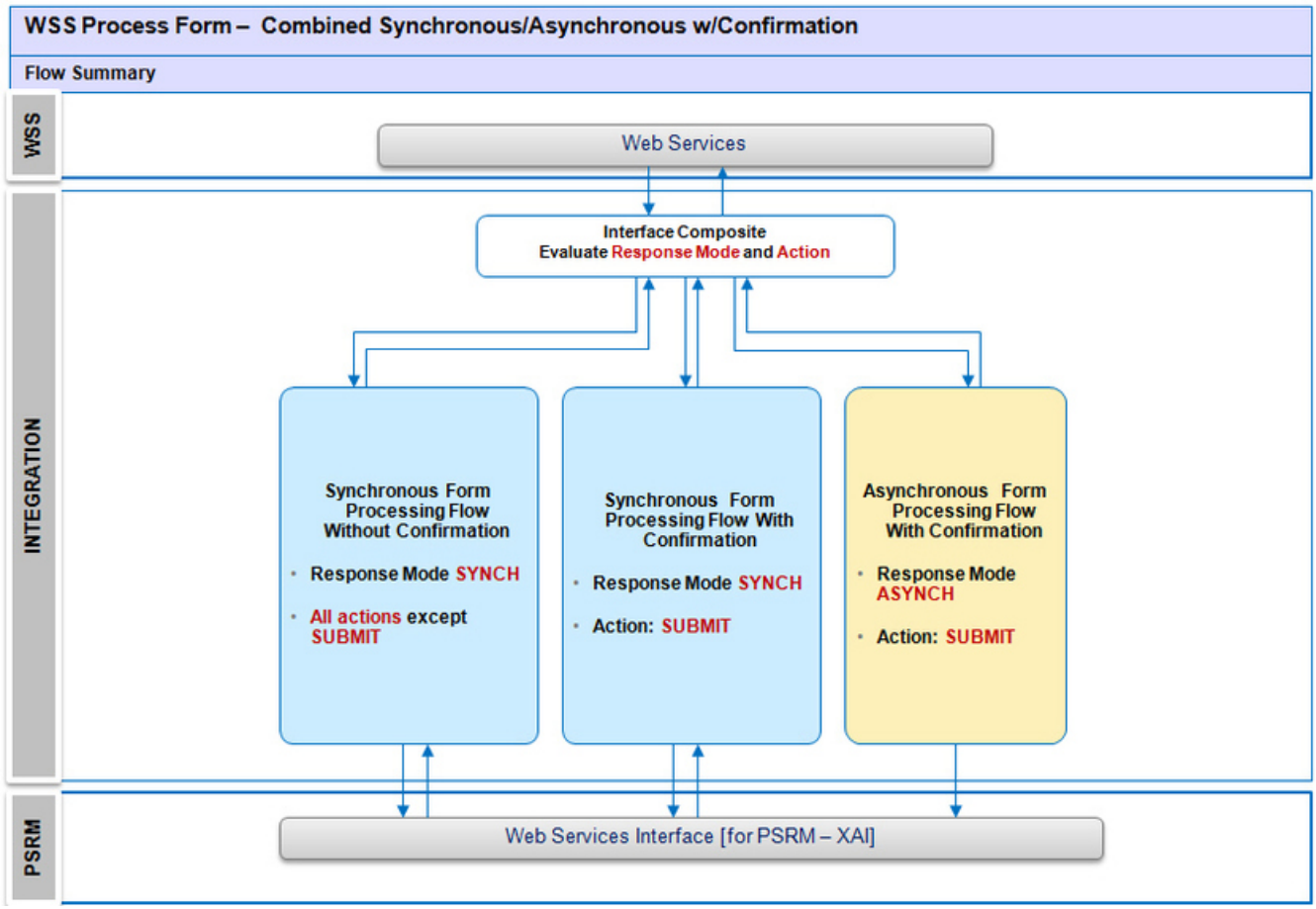


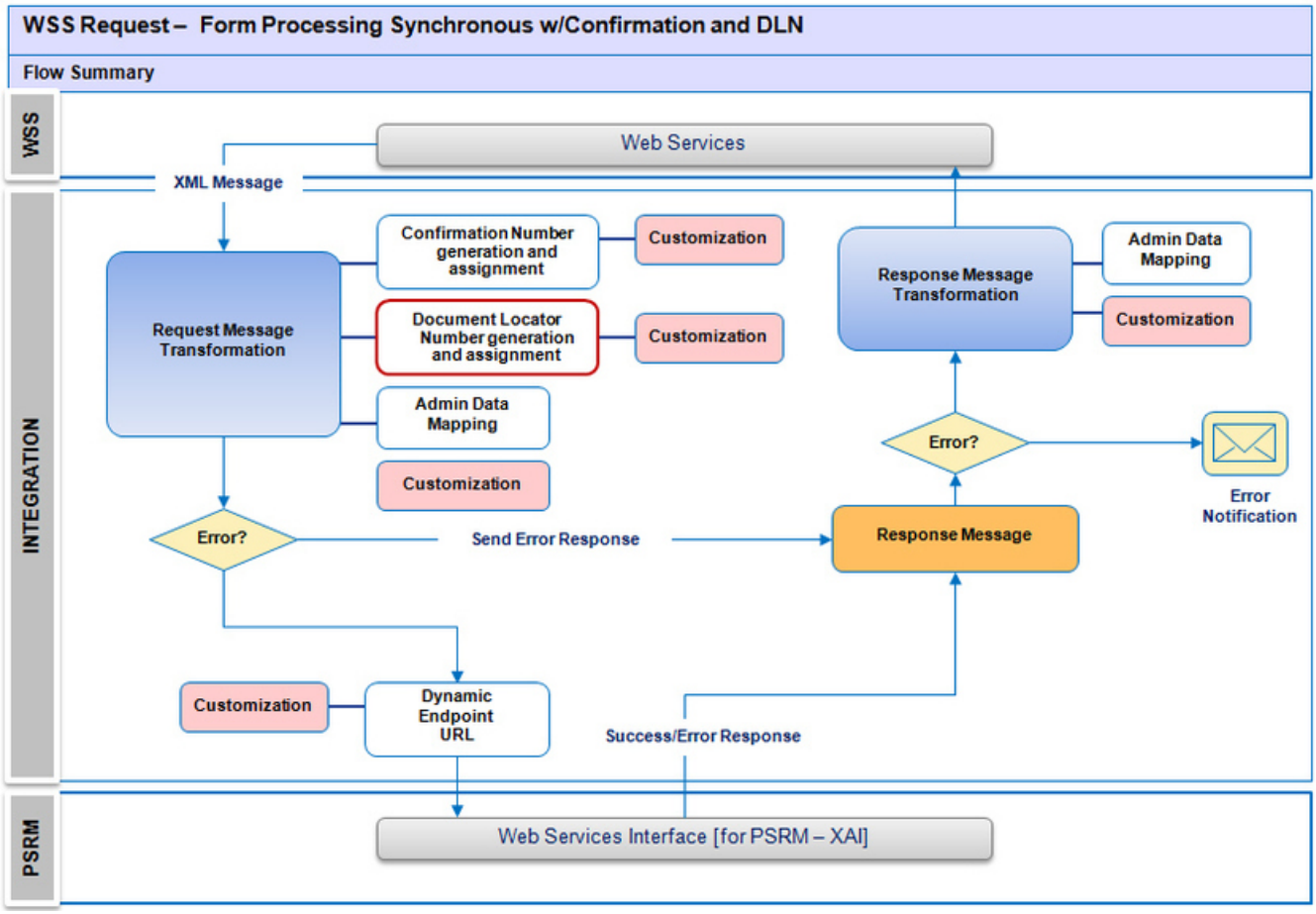
Form Process Flows

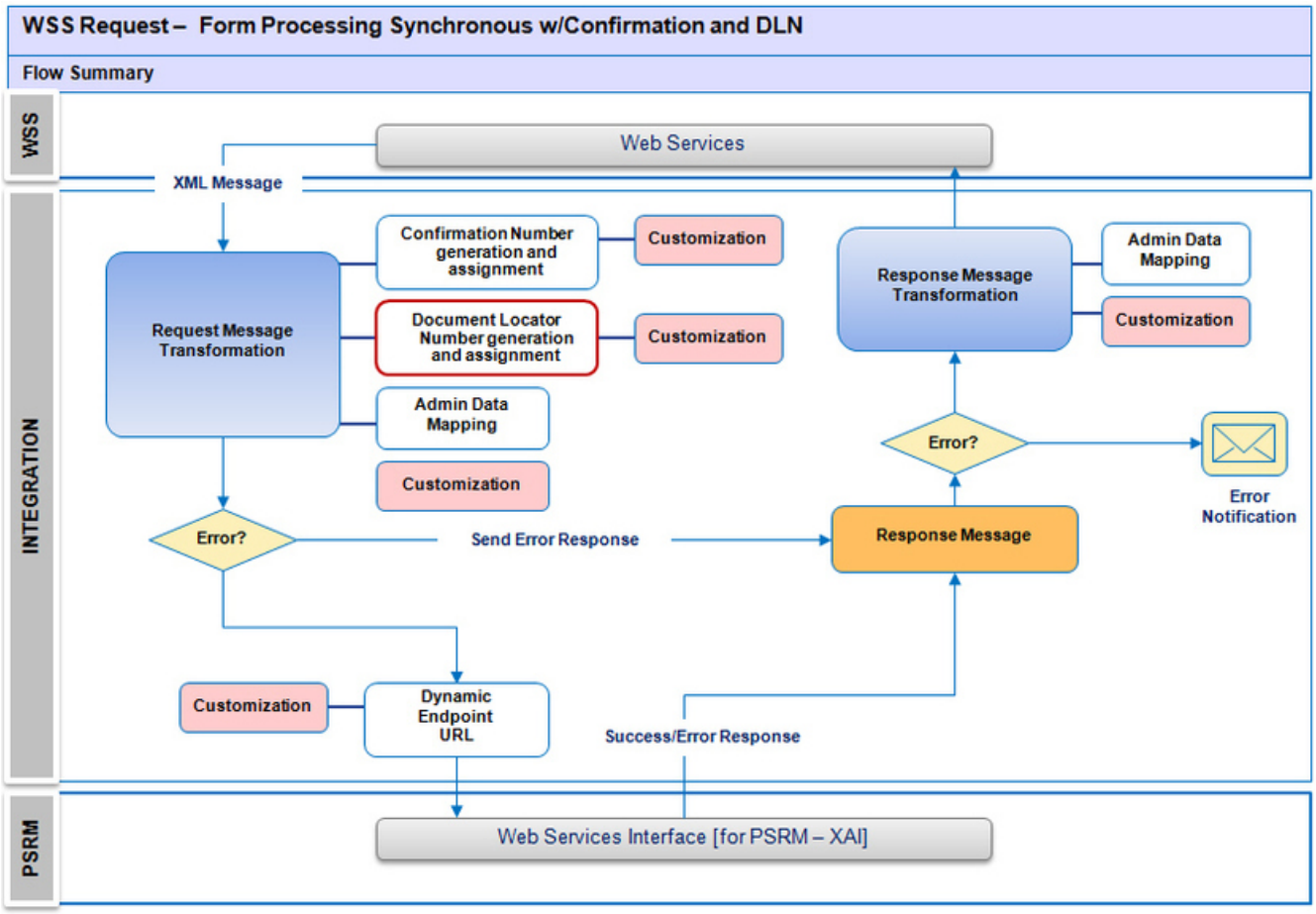
This integration includes special flows that provide interaction with solution components as well as the revenue management application.

The **Form Process** flow is a combination of synchronous and asynchronous requests controlled by an interface process and is dependent on the action specified on the request. Similar flows are created for Tax and Business Registration Forms.

Form submission includes special logic for **Document Locator Number** creation.

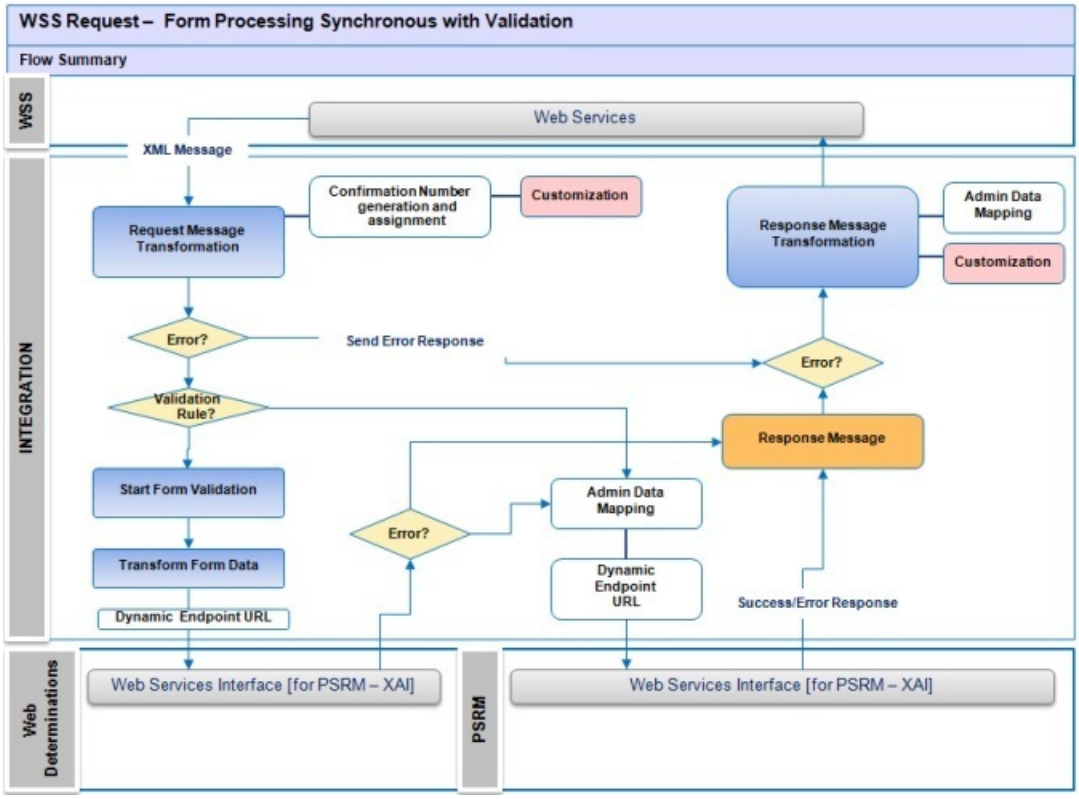

WSS Process Form – Combined Synchronous/Asynchronous w/Confirmation
Flow Summary
WSS
Web Services
INTEGRATION
Interface Composite
Evaluate Response Mode and Action
Synchronous Form Processing Flow Without Confirmation
Response Mode SYNCH
All actions except SUBMIT
Synchronous Form Processing Flow With Confirmation
Response Mode SYNCH
Action: SUBMIT
Asynchronous Form Processing Flow With Confirmation
Response Mode ASYNCH
Action: SUBMIT
PSRM
Web Services Interface [for PSRM – XAI]


WSS Request – Form Processing Synchronous w/Confirmation and DLN
Flow Summary
WSS
Web Services
XML Message
INTEGRATION
Request Message Transformation
Confirmation Number generation and assignment
Customization
Document Locator Number generation and assignment
Customization
Admin Data Mapping
Customization
Error?
Send Error Response
Customization
Dynamic Endpoint URL
Response Message Transformation
Admin Data Mapping
Customization
Error?
Error Notification
Response Message
Success/Error Response
PSRM
Web Services Interface [for PSRM – XAI]

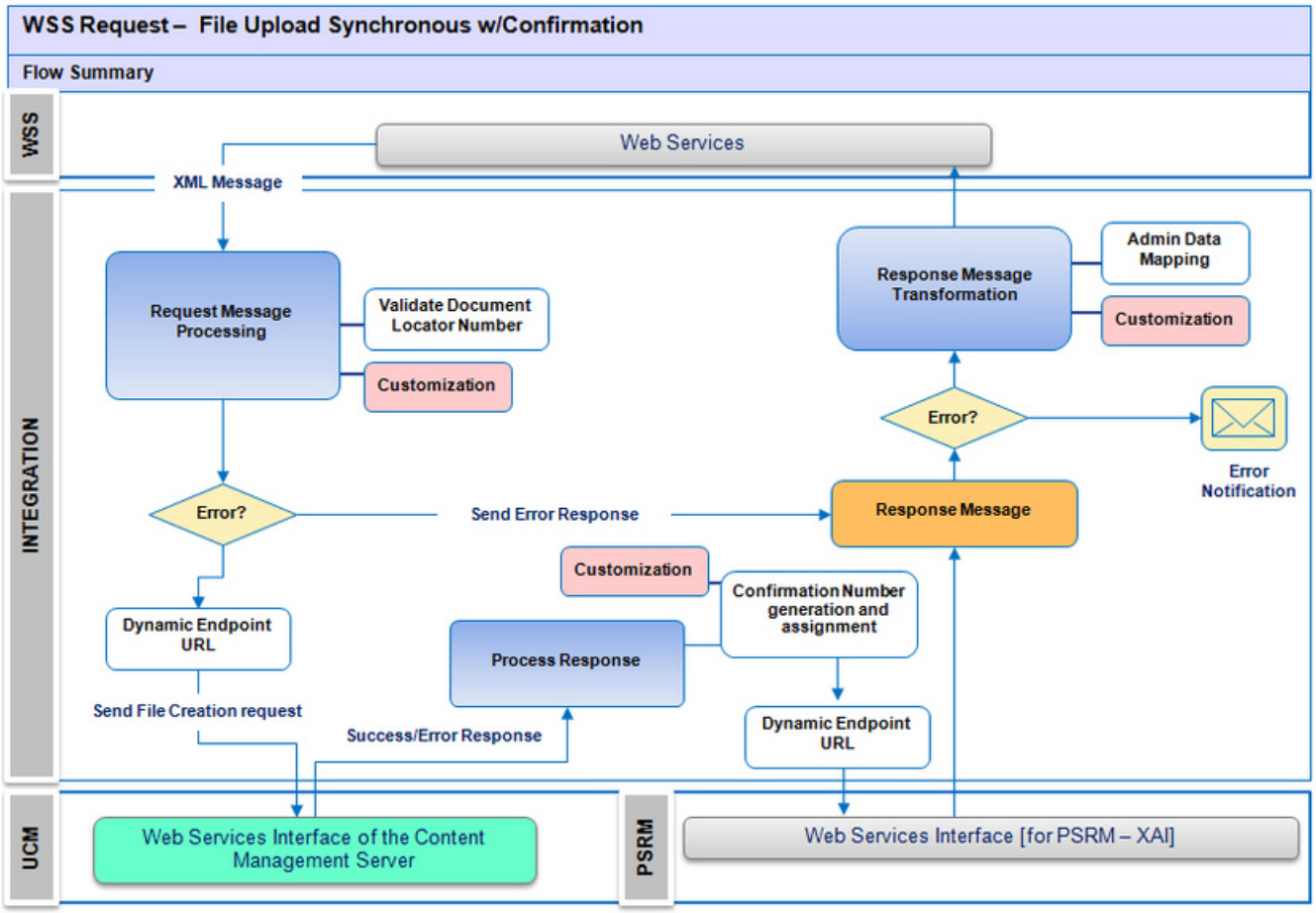


Form validation is performed first through web service interaction with the rulebase installed on the Oracle Determination Server. The rulebase name and Oracle Determination Service location are specified on the request. Form data is transformed into a rulebase generic web service request format. The response from the rulebase contains validation errors and updated form data. The response from the rulebase is transformed back into the original form data XML format. If no errors are detected, the request is forwarded to the revenue management system for additional validation.

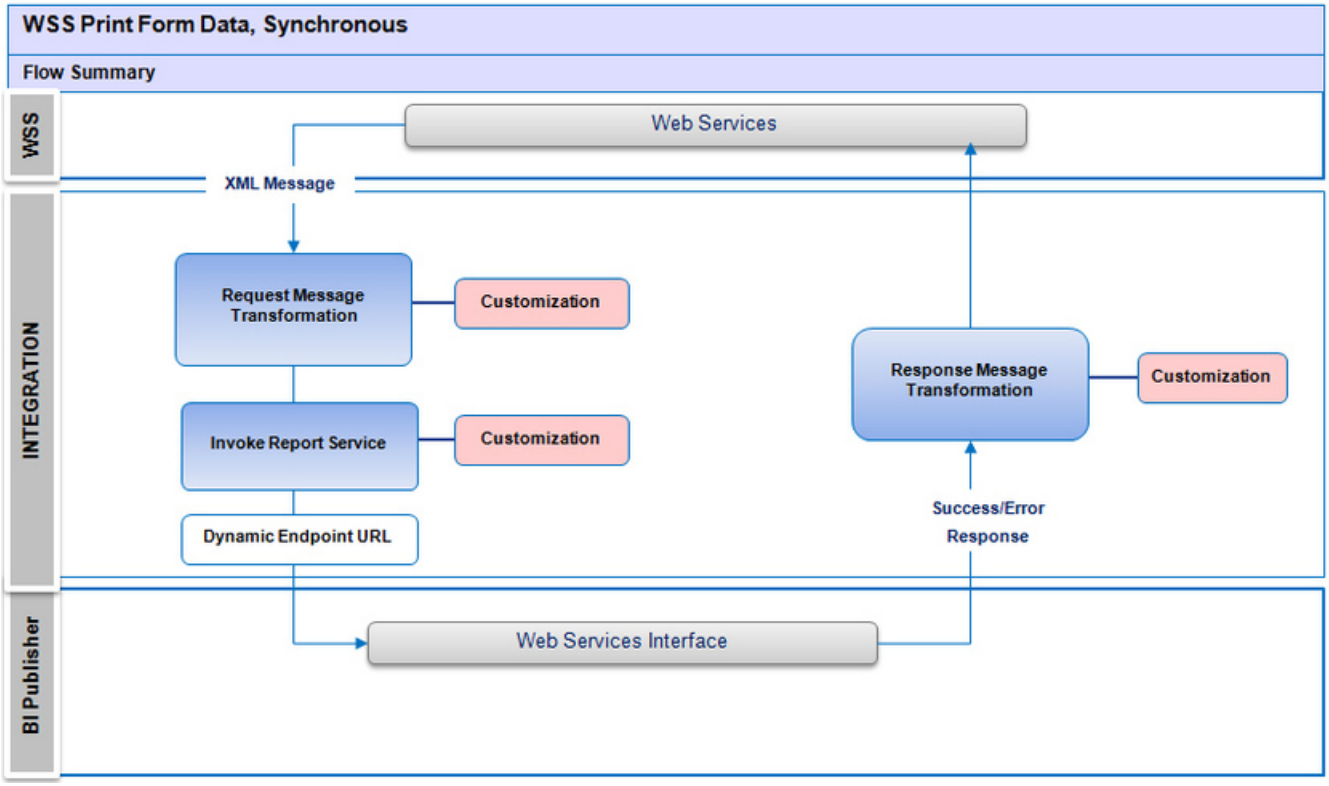
The final response from the revenue management system contains updated form data and/or errors.



The **Supporting Documents (File) Upload** process generates a unique file name and invokes a web service in order to store the file. The integration with Oracle Universal Content Management is provided out-of-box, but this flow may be customized to use another file storage/document management server. The web service response is expected to contain the document's URL, and this URL is populated on the request that is forwarded to the revenue management system for tracking purposes. The final response is expected to contain the confirmation message.



The **Form Print** process flow is synchronous; it invokes a document generation sub-flow passing the form data as well-formed XML. The out-of-box integration includes a sub-flow that uses the Oracle Business Intelligence Publisher's ReportService. It sends form data with the request and receives the printable document content. The response is sent back to the Self Service Portal.



Integration Solution Flows

Flow Name	Description	Integration Flow Pattern
Generic Taxpayer Service Request	Self-service application initiated	Combination of sync and async process with limited response (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	Self-service application initiated	Async flow with no response
Report Reconciliation	Self-service application initiated	Async flow with no response
Retrieve Payments due	Self-service application initiated	Sync flow without confirmation id
Enrollment Request	Self-service application initiated	Combination of sync and async process with limited response (confirmation id), enrollment id generation and user access store update and Oracle BPM Worklist integration
User Enrollment Query/Refresh	Self-service application initiated	Sync flow without confirmation id, with user access store inquiry and update

Flow Name	Description	Integration Flow Pattern
User Enrollment Summary	Self-service application initiated	Sync flow without confirmation id, with user access store inquiry
Tax Form Process	Self-service application initiated	Combination of sync and async process with limited response (confirmation id), document locator id generation and Oracle Determination Server integration
Business Registration Form Process	Self-service application initiated	Combination of sync and async process with limited response (confirmation id), document locator id generation and Oracle Determination Server integration
Supporting Document (File) Upload	Self-service application initiated	Sync flow with confirmation id and Oracle Universal Content Management integration
Retrieve Active Form Types	Self-service application initiated	Sync flow without confirmation id
Refresh Lookup	Self-service application initiated	Sync flow without confirmation id
Import Form Types	Self-service application initiated	Sync flow without confirmation id
Account Summary	Self-service application initiated	Sync flow without confirmation id
Account Filing History	Self-service application initiated	Sync flow without confirmation id
Account Payment History	Self-service application initiated	Sync flow without confirmation id
Account Alerts	Self-service application initiated	Sync flow without confirmation id
Taxpayer Summary	Self-service application initiated	Sync flow without confirmation id
Taxpayer Contact Info	Self-service application initiated	Sync flow with optional confirmation id
Taxpayer Correspondence Info	Self-service application initiated	Sync flow without confirmation id
Address Maintenance	Self-service application initiated	Sync flow with confirmation id

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

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	PSRM XML Fragment
<confirmation> <confirmHeader> <messageNumber/> <messageParameters> <parameters type="list"> <parameterType/> <parameterValue/> </parameters> </messageParameters> <currency/> <messageTxtOvrd/> </confirmaHeader>	<confirmation> <confirmHeader> <messageCategory/> <messageNumber/> <messageParameters> <parameters type="list"> <parameterType/> <parameterValue/> </parameters> </messageParameters> <currency/> <messageTxtOvrd/>

OTSS XML Fragment	PSRM XML Fragment
<confirmDetails type=”list”> <messageNumber/> <messageParameters> <parameters type="list"> <parameterType/> <parameterValue/> </parameters> </messageParameters> <currency/> <messageTxtOvrd/> </confirmDetails> </confirmation>	</confirmaHeader> <confirmDetails type=”list”> <messageCategory/> <messageNumber/> <messageParameters> <parameters type="list"> <parameterType/> <parameterValue/> </parameters> </messageParameters> <currency/> <messageTxtOvrd/> </confirmDetails> </confirmation>

In the absence of a DVM value setup for Message Number in the Confirmation Header and Confirmation Detail, the replacement values are taken from the ConfigurationProperties.xml file (see *Module Configurations*).

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 see *Module Configurations*).
- 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'.
- The element <messageRef> on the <errorMessage> XML fragment contains the original message number/category number and message category code indicating the source of the message (self-service portal, revenue management, or BPEL). This is be populated by the BPEL composite.

OTSS XML Fragment	PSRM XML Fragment
<status/> <errorMsg> <messageNumber/> <messageParameters> <parameters type="list"> <parameterType/> <parameterValue/> </parameters> </messageParameters> <currency/> <messageTxtOvrd/> <messageRef/> </errorMsg>	<status/> <errorMsg> <messageCategory/> <messageNumber/> <messageParameters> <parameters type="list"> <parameterType/> <parameterValue/> </parameters> </messageParameters> <currency/> <messageTxtOvrd/> <messageRef/> </errorMsg>

Failed Message Delivery

Confirmation information is included in multiple messages.

Under normal circumstances, the web service 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*).

OTSS XML Fragment	PSRM XML Fragment

`<confirmationId/>`
`...`
`<status/>`
`<errorMsg>`
`<messageNumber/>`
`<messageParameters>`
`<parameters type="list">`
`<parameterType/>`
`<parameterValue/>`
`</parameters>`
`</messageParameters>`
`<currency/>`
`<messageTxtOvrd/>`
`</errorMsg>`
	`<confirmationId/>`
	`...`
	`<status/>`
	`<errorMsg>`
	`<messageCategory/>`
	`<messageNumber/>`
	`<messageParameters>`
	`<parameters type="list">`
	`<parameterType/>`
	`<parameterValue/>`
	`</parameters>`
	`</messageParameters>`
	`<currency/>`
	`<messageTxtOvrd/>`
	`</errorMsg>`

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.

Domain Value Mappings

Name	Integration Points	Purpose
OTSS_MessageNumber	All	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 **Columns:**

Name	Integration Points	Purpose
		• OTSS_MessageNumber • EXT_MessageCategoryNumber
OTSS_ MessageParameterType	All	Maps revenue management **message parameter type** to the self-service application **message parameter type**. **Columns:** • OTSS_MessageParmType Supported delivered values: **DATE**, **CURRENCY**, **STRING**, **NUMBER** • EXT_MessageParmType
OTSS_Currency	All	Maps revenue management **currency code** to self-service application **currency code**. **Columns:** • OTSS_CurrencyCode Delivered default value: **USD** • EXT_CurrencyCode
OTSS_ServiceRequestType	All service request-based integration points	Maps revenue management **service request type** to self-service application **service task type**. **Columns:** • OTSS_SRType Initially delivered values: supported service request types • EXT_SRType
OTSS_Field Codes	All	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. **Columns:** • OTSS_FieldNameValue **Format:** **[FIELD NAME][separator][VALUE]**, where the separator character is defined in ConfigurationProperties.xml • EXT_FieldValue
OTSS_PaymentType	All payment-related integration points	Maps revenue management **payment** [tender] **type** to self-service application payment type (values of the lookup **TENDERTYPE**). **Columns:** • OTSS_PaymentTypeCode Initially delivered values: **CHECKING**, **SAVING**, **CREDIT** • EXT_PaymentTypeCode
OTSS_PaymentDestination	All payment-related integration points	Maps revenue management **payment destination** to self-service application **payment destination**. **Columns:** • OTSS_PaymentDestValue Initially delivered values: supported payment destinations • EXT_PaymentDestValue

Name	Integration Points	Purpose
OTSS_PaymentVendor	All payment-related integration points	Maps revenue management **payment vendor** to self-service application **payment provider**. **Columns:** • OTSS_PaymentVendor • EXT_PaymentVendor
OTSS_FeeRequirement	All payment-related integration points	Maps a combination of revenue management **payment vendor**, **payment destination**, and **fee requirement** combination to self service **application fee requirement**. **Columns:** • OTSS_FeeRequirement • EXT_FeeReq_PayVendor_PayDestination **Format:** **[FEE REQUIREMENT][separator]PAYMENT VENDOR][separator][PAYMENT DESTINATION]** where the [separator] character is defined in ConfigurationProperties.xml
OTSS_Country	All payment-related integration points	Maps revenue management **country codes** to self-service and other applications **country codes**. **Columns:** • OTSS_Country • EXT_Country
OPC_PaymentType	Official Payments Corp integration points	Maps Official Payments Corp. **payment**[account]**type** code to revenue management **payment** [tender] **type**. **Columns:** • OPC_PaymentType Valid values: **V, VISA, M, MC, A, AMEX, D, DISC, ES, EC, EBS, EBC** • EXT_PaymentType
OPC_PaymentCountry	Official Payments Corp integration points	Maps Official Payments Corp. **payment**[account]**type** code to revenue management **payment** [tender] **type**. Maps OPC country code to revenue management country code.
OTSS_UserAccessStatus		• OTSS_AccessStatus Valid values: A, P, I, H • EXT_AccessStatus
OTSS_AccessType		• OTSS_AccessType Valid values: TAXROLE • EXT_AccessType
OTSS_AlertType		• OTSS_AlertType Valid values: OPEN_COLLECTION, OVERDUE_BALANCE, RETURN_MISSING, TAXPAYER_INFO • EXT_AlertType

Name	Integration Points	Purpose
OTSS_AddressType		• OTSS_AddressType Valid values: MAILING • EXT_AddressType
OTSS_ AddressChangeReason		• OTSS_AddressChangeReason • EXT_AddressChangeReason
OTSS_FilingStatus		• OTSS_FilingStatus Valid values: PAYMENT_DUE, INPROGRESS, COMPLETE, NOFILING • EXT_FilingStatus
OTSS_PaymentStatus		• OTSS_PaymentStatus Valid values: COMPLETE, ISSUE_DETECTED • EXT_PaymentStatus
OTSS_PhoneType		• OTSS_PhoneType • EXT_PhoneType
OTSS_TaxType		• OTSS_TaxType • EXT_TaxType
OTSS_TaxpayerType		• OTSS_TaxpayerType • EXT_TaxpayerType
OTSS_FormAction		• OTSS_FormAction Valid values: SUBMIT, COPY, READY, VALIDATE • EXT_FormAction
OTSS_FormCategory		• OTSS_FormCategory Valid values: REGFORM, TAXFORM • EXT_FormCategory
OTSS_FormDataType		• OTSS_FormDataType Valid values: BOOLEAN, TEXT, DATETIME, DATE, LOOKUP, CURRENCY, NUMBER, TIME • EXT_FormDataType
OTSS_ FormSectionOccurence		• OTSS_Occurence Valid values: SINGLE, UNLIMITED • EXT_FormSectionOccurence
OTSS_LineOfBusiness		• OTSS_LOB Valid values: SINGLE, UNLIMITED • EXT_LOB

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-PSRM/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.
ConfimationHeader.Default.MessageNumber	100	Default value for the message number in the Confirmation header.
Default.ErrorNumber	101	Default Error number
Default.FailureNumber	102	Default Failure number.
ConfimationDetails.Default.MessageNumber	104	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.

Configuration Property	Default Value	Description
		In this table the integration flow **OTSSGetExternalPaymentDataEBF** stores external payment transaction staging records. The property is set during the installation check.
OTSS.UserAccess.DataSource		jdbc/OTSS-USRACS
ConfirmationHeader.MessageDelivered.MessageNumber	105	The message number used by SOA composites as a confirmation header when they need to communicate successful request processing.
ConfirmationDetails.Default.MessageNumber	104	The message number used by SOA composites for confirmation details when they need to communicate successful request processing.
MessageCategoryNumber.Separator	:	
System.UserName	SYSTEM	This is a user name specified on the payment post-back messages originating from the external Payment Vendor. Used when processing **OTSSPaymentPostingRequestEBF**.
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
ResultCode.EmailContent	ResultCode value is not equal to A	

Configuration Property	Default Value	Description
		payment in error. In this scenario the post-back XML is not processed any further.
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	
DLN.Size	16	
DLN.Action.Create	CREATE	
DLN.Action.Validate	VALIDATE	
EnrollmentStatus.Blocked	B	
EnrollmentStatus.Hold	H	
EnrollmentStatus.Approved	A	
Enrollment.UserAccessService.Name	{http://xmlns.oracle.com/OTSS/Industry/Tax/OTSSUserAccessService}OTSSUserAccessProcessorService	
Enrollment.UserAccessService.Port	OTSSUserAccessService_pt	
Enrollment.UserAccessService.URL	.../soa-infra/services/OTSS-PSRM/OTSSUserAccessService/OTSSUserAccessProcessorService	Updated with the actual endpoint URL during the installation

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

Configuration Property	Default/Delivered Value	Description
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/ OTSSConfirmationIdBPELProcess}otssconfirmationidbpelprocess_client_ep	
RequestStatusInquiry.ConfirmationNumberService.Port	OTSSConfirmationIdBPELProcess_pt	
RequestStatusInquiry.ConfirmationNumberService.URL	**.../ soa-infra/services/OTSS-PSRM/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	

Configuration Property	Default/Delivered Value	Description
OneTimePayment.ConfirmationNumberService.URL	...soa-infra/services/OTSS-PSRM/ 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 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-PSRM/ 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

Configuration Property	Default/Delivered Value	Description
		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**
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-PSRM/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.

Configuration Property	Default/Delivered Value	Description
ExternalPaymentData.Service.Name	{http://ouaf.oracle.com/spl/XAIXapp/xaiserver/ TSPrepareExtPaymentData} TSPrepareExtPaymentDataService	
ExternalPaymentData.Port.Name	TSPrepareExtPaymentDataPortType	
ExternalPaymentData.Endpoint.URL	.../TSPrepareExtPaymentData	**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/ TSOneTimePayment}TSOneTimePaymentService	
PaymentPostBack.Port.Name	TSOneTimePaymentPort	
PaymentPostBack.Endpoint.URL	.../TSOneTimePayment	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
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

Configuration Property	Default/Delivered Value	Description
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	.../TSProcessExtPayReportRecord	**Updated with the actual URL during the installation**
Extension.PreXform	**false**	
Service Name: **OTSSRetrievePaymentsDueEBF**		
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
RetrievePaymentDue.Port.Name	TSRetrievePaymentsDuePort	
RetrievePaymentDue.Service.Name	{http://ouaf.oracle.com/spl/XAIXapp/xaiserver/TSRetrievePaymentsDue}TSRetrievePaymentsDueService	
RetrievePaymentDue.Endpoint.URL	../TSRetrievePaymentsDue	**Updated with the actual URL during the installation**
FieldValue.Separator	:	
Service Name: **OTSSRetrieveActiveFormTypesEBF**		
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
RetrieveActiveFormTypes.Service.Name	{http://ouaf.oracle.com/spl/XAIXapp/xaiserver/TSRetrieveActiveFormTypes}TSRetrieveActiveFormTypesService	
RetrieveActiveFormTypes.Port.Name	TSRetrieveActiveFormTypesPort	
RetrieveActiveFormTypes.Endpoint.URL	.../ TSRetrieveActiveFormTypes	**Updated with the actual endpoint URL during the installation**
Service Name: **OTSSRetrieveFormTypeDefinitionsEBF**		

Configuration Property	Default/Delivered Value	Description
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
RetrieveFormTypeDefinitions.Service.Name	{http://ouaf.oracle.com/spl/XAIXapp/xaiserver/TSRetrieveFormTypeDefinitions}TSRetrieveFormTypeDefinitions	
RetrieveFormTypeDefinitions.Port.Name	TSRetrieveFormTypeDefinitions	
RetrieveFormTypeDefinitions.Endpoint.URL	.../ TSRetrieveFormTypeDefinitions	**Updated with the actual endpoint URL during the installation**
Service Name: **OTSSRefreshFormLookupEBF**		
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
RefreshFormLookup.Service.Name	{http://ouaf.oracle.com/spl/XAIXapp/xaiserver/**RefreshFormLookup**}TS**RefreshFormLookup**	
RefreshFormLookup.Port.Name	TS**RefreshFormLookup**	
RefreshFormLookup.Endpoint.URL	.../ TS**RefreshFormLookup**	**Updated with the actual endpoint URL during the installation**
Service Name: **OTSSGetTaxAccountSummaryEBF**		
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

Configuration Property	Default/Delivered Value	Description
GetTaxAccountSummary.Service.Name	{http://ouaf.oracle.com/spl/XAIXapp/xaiserver/TS GetTaxAccountSummary } TSGetTaxAccountSummaryService	
GetTaxAccountSummary.Port.Name	TSGetTaxAccountSummary Port	
GetTaxAccountSummary.Endpoint.URL	.../TS GetTaxAccountSummary	**Updated with the actual endpoint URL during the installation**
Service Name: **OTSSGetTaxAccountAlertsEBF**		
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
GetTaxAccountAlerts.Service.Name	{http://ouaf.oracle.com/spl/XAIXapp/xaiserver/TS GetTaxAccountAlerts } TSGetTaxAccountAlerts Service	
GetTaxAccountAlerts.Port.Name	TSGetTaxAccountAlerts Port	
GetTaxAccountAlerts.Endpoint.URL	.../TS GetTaxAccountAlerts	**Updated with the actual endpoint URL during the installation**
Service Name: **OTSSGetPaymentHistoryEBF**		
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
GetPaymentHistory.Service.Name	{http://ouaf.oracle.com/spl/XAIXapp/xaiserver/TS GetPaymentHistory } TSGetPaymentHistoryService	
GetPaymentHistory.Port.Name	TSGetPaymentHistoryPort	
GetPaymentHistory.Endpoint.URL	.../TSGetPaymentHistory	**Updated with the actual endpoint URL during the installation**
Service Name: **OTSSGetFilingHistoryEBF**		
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

Configuration Property	Default/Delivered Value	Description
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
GetFilingHistory.Service.Name	{http://ouaf.oracle.com/spl/XAIXapp/ xaiserver/TS GetFilingHistory } TSGetFilingHistoryService	
GetFilingHistory.Port.Name	TSGetFilingHistoryPort	
GetFilingHistory.Endpoint.URL	.../TSGetFilingHistory	**Updated with the actual endpoint URL during the installation**
Service Name: OTSSGetTaxpayerSummaryEBF		
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
GetTaxPayerSummary.Port.Name	TSGetTaxPayerSummaryPort	
GetTaxPayerSummary.Service.Name	{http://ouaf.oracle.com/spl/XAIXapp/ xaiserver/ TSGetTaxPayerSummary}TSGetTaxPayerSummaryService	
GetTaxPayerSummary.Endpoint.URL	../TSGetTaxPayerSummary	**Updated with the actual endpoint URL during the installation**
Service Name: OTSSGetTaxpayerCorrespondenceInfoEBF		
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
GetTaxpayerCorrespondenceInfo.Port.Name	TSGetTaxpayerCorrespondenceInformationPort	
GetTaxpayerCorrespondenceInfo.Service.Name	{http://ouaf.oracle.com/spl/XAIXapp/ xaiserver/ TSGetTaxpayerCorrespondenceInformation} TSGetTaxpayerCorrespondenceInformationService<	
GetTaxpayerCorrespondenceInfo.Endpoint.URL	../ TSGetTaxpayerCorrespondenceInformation	**Updated with the actual endpoint URL during the installation**

Configuration Property	Default/Delivered Value	Description
Service Name OTSSGetTaxpayerContactInfoEBF		
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
GetTaxpayerContactInfo.Port.Name	TSGetTaxpayerContactInformationPort	
GetTaxpayerContactInfo.Service.Name	{http://ouaf.oracle.com/spl/XAIXapp/ xaiserver/ TSGetTaxpayerCorrespondenceInformation} TSGetTaxpayerContactInformationService<	
GetTaxpayerContactInfo.Endpoint.URL	../ TSGetTaxpayerContactInformation	**Updated with the actual endpoint URL during the installation**
GetTaxpayerContactInfo.ConfirmationNumberService.Name	{http://xmlns.oracle.com/OTSS/ OTSSConfirmationIdService/ OTSSConfirmationIdBPELProcess} otssconfirmationidbpelprocess_client_ep	
GetTaxpayerContactInfo.ConfirmationNumberService.Port	OTSSConfirmationIdBPELProcess_pt	
GetTaxpayerContactInfo.ConfirmationNumberService.URL	../soa-infra/services/OTSS-PSRM/ OTSSConfirmationIdService/ otssconfirmationidbpelprocess_client_ep	**Updated with the actual endpoint URL during the installation**
GetTaxpayerContactInfo.Prefix	CIID	
GetTaxpayerContactInfo.Action.Update	UPDATE	
Service Name: OTSSAddressMaintenanceEBF		
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
AddressMaintenance.Port.Name	TSAddressMaintenancePort	
AddressMaintenance.Service.Name	{http://ouaf.oracle.com/spl/XAIXapp/ xaiserver/ TSAddressMaintenance}TSAddressMaintenanceService	
AddressMaintenance.Endpoint.URL	http://<server_name>:7351/spl/ XAIApp/xaiserver/TSAddressMaintenance	
AddressMaintenance.ConfirmationNumberService.Name	http://xmlns.oracle.com/OTSS/ OTSSConfirmationIdService/ OTSSConfirmationIdBPELProcess}	

Configuration Property	Default/Delivered Value	Description
	otssconfirmationidbpelprocess_client_ep	
AddressMaintenance.ConfirmationNumberService.Port	OTSSConfirmationIdBPELProcess_pt	
AddressMaintenance.ConfirmationNumberService.URL	http://<server_name>:8051/soa-infra/ services/ OTSS-PSRM/OTSSConfirmationIdService/ otssconfirmationidbpelprocess_client_ep	
AddressMaintenance.Prefix	AMID	
Service Name: OTSSProcessRegistrationFormServiceProvider		
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
ProcessRegistrationForm.Endpoint.URL	../TSProcessRegistrationForm	**Updated with the actual endpoint URL during the installation**
Service Name: OTSSProcessRegistrationFormEBF		
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
ProcessRegistrationForm.Endpoint.URL	http://<server_name>:7351/spl/XAIApp/ xaiserver/TSProcessRegistrationForm	
RegistrationFormSequence.Prefix	RFID	
RegistrationForm.ConfirmationNumberService.Name	{http://xmlns.oracle.com/OTSS/ OTSSConfirmationIdService/ OTSSConfirmationIdBPELProcess} otssconfirmationidbpelprocess_client_ep	
RegistrationForm.ConfirmationNumberService.Port	OTSSConfirmationIdBPELProcess_pt	
RegistrationForm.ConfirmationNumberService.URL	../soa-infra/services/OTSS-PSRM/ OTSSConfirmationIdService/ otssconfirmationidbpelprocess_client_ep	**Updated with the actual endpoint URL during the installation**
ProcessRegistrationForm.RequestMode.Sync	SYNCH	
ProcessRegistrationForm.RequestMode.Async	ASYNCH	
FieldValue.Separator	:	
RegistrationForm.DLN.Prefix	RFDLN	
RegistrationForm.DLNService.Name	{http://xmlns.oracle.com/OTSS/ OTSSDocumentLocatorNumberService/ OTSSDocumentLocatorNumberBPELProcess} otssdocumentlocatornumberbpelprocess_ client_ep	
RegistrationForm.DLNService.Port	OTSSDocumentLocatorNumberBPELProcess_ pt	

Configuration Property	Default/Delivered Value	Description
RegistrationForm.DLNService.URL	.../soa-infra/services/OTSS-PSRM/ OTSSDocumentLocatorNumberService/ otssdocumentlocatornumberbpelprocess_ client_ep	**Updated with the actual endpoint URL during the installation**
RegistrationForm.Action.Submit	SUBMIT	
RegistrationForm.Action.Validate	VALIDATE	
RegistrationForm.Action.Ready	READY	
RegistrationForm.BackEndValidation.Enabled	true	
RegistrationForm.FormValidationService.Name	{http://xmlns.oracle.com/OTSS/ OTSSFormValidationService/ OTSSFormValidationBPELProcess} OTSSFormValidationBPELProcess_client_ ep	
RegistrationForm.FormValidationService.Port	OTSSFormValidationBPELProcess_pt	
RegistrationForm.FormValidationService.URL	../soa-infra/services/OTSS-PSRM/ OTSSFormValidationService/ otssformvalidationbpelprocess_client_ep	**Updated with the actual endpoint URL during the installation**
Service Name: OTSSProcessTaxFormServiceProvider		
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
ProcessTaxForm.Endpoint.URL	../TSProcessTaxForm	**Updated with the actual endpoint URL during the installation**
Service Name: **OTSSProcessTaxFormEBF**		
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
ProcessTaxForm.Endpoint.URL	http://<server_name>:7351/spl/XAIApp/ xaiserver/TSProcessTaxForm	
TaxFormSequence.Prefix	TFID	
TaxForm.ConfirmationNumberService.Name	{http://xmlns.oracle.com/OTSS/ OTSSConfirmationIdService/ OTSSConfirmationIdBPELProcess} otssconfirmationidbpelprocess_client_ep	
TaxForm.ConfirmationNumberService.Port	OTSSConfirmationIdBPELProcess_pt	
TaxForm.ConfirmationNumberService.URL	../soa-infra/services/OTSS-PSRM/ OTSSConfirmationIdService/ otssconfirmationidbpelprocess_client_ep	**Updated with the actual endpoint URL during the installation**
ProcessTaxForm.RequestMode.Sync	SYNCH	
ProcessTaxForm.RequestMode.Async	ASYNCH	
FieldValue.Separator	:	

Configuration Property	Default/Delivered Value	Description
TaxForm.DLN.Prefix	TFDLN	
TaxForm.DLNService.Name	{http://xmlns.oracle.com/OTSS/ OTSSDocumentLocatorNumberService/ OTSSDocumentLocatorNumberBPELProcess} otssdocumentlocatornumberbpelprocess_ client_ep	
TaxForm.DLNService.Port	OTSSDocumentLocatorNumberBPELProcess_ pt	
TaxForm.DLNService.URL	.../soa-infra/services/OTSS-PSRM/ OTSSDocumentLocatorNumberService/ otssdocumentlocatornumberbpelprocess_ client_ep	**Updated with the actual endpoint URL during the installation**
TaxForm.Action.Submit	SUBMIT	
TaxForm.Action.Validate	VALIDATE	
TaxForm.Action.Ready	READY	
TaxForm.BackEndValidation.Enabled	true	
TaxForm.FormValidationService.Name	{http://xmlns.oracle.com/OTSS/ OTSSFormValidationService/ OTSSFormValidationBPELProcess} OTSSFormValidationBPELProcess_client_ ep	
TaxForm.FormValidationService.Port	OTSSFormValidationBPELProcess_pt	
TaxForm.FormValidationService.URL	../soa-infra/services/OTSS-PSRM/ OTSSFormValidationService/ otssformvalidationbpelprocess_client_ep	**Updated with the actual endpoint URL during the installation**
Service Name: OTSSPrintFormEBF		
Extension.PreXformOTSS	false	If set to **true** the request pre-transformation extension
Extension.PostXformOTSS	false	If set to **true** the request pre-transformation extension
DocumentService.Port.Name	ReportService	
DocumentService.Service.Name	{http://xmlns.oracle.com/oxp/service/ v2}ReportService	
DocumentService.Endpoint.URL	.../xmlpserver/services/v2/ReportService	**BI Publisher Server Location**
DocumentService.Username		**User name for document service**
DocumentService.Password		**Password for document service.**
DocumentService.ReportLocation	/~weblogic/Drafts/	
DocumentService.ReportName	<FormCategory name="TAXFORM">GenericFormData.xdo</ FormCategory> <FormCategory name="REGFORM">GenericFormData.xdo</ FormCategory>	
DocumentService.DocumentFormat	pdf	
PrintFormFailed.ErrorNumber	150	
ReportData.Transformer		**Will be populated by customization**
EncodingRequired.Flag	false	
PrintForm.EncodeDataService.Name	{http://xmlns.oracle.com/OTSS/Industry/Tax/ OTSSBase64EncodeDataService} OTSSBase64EncodedDataGeneratorService	

Configuration Property	Default/Delivered Value	Description
PrintForm.EncodeDataService.Port	OTSSBase64EncodeDataService_pt	
PrintForm.EncodeDataService.URL	.../soa-infra/services/OTSS-PSRM/ OTSSBase64EncodeDataService/ OTSSBase64EncodedDataGeneratorService	**Updated with the actual endpoint URL during the installation**
Service Name OTSSFormValidationService		
GenericRequest.Path	/assess/soap/generic/	
Service Name: OTSSFormUploadEBF		
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
FileUpload.Service.Name	{http://ouaf.oracle.com/spl/XAIXapp/ xaiserver/ TSUploadSupportingDocument} TSUploadSupportingDocumentService	
FileUpload.Port.Name	TSUploadSupportingDocumentPort	
FileUpload.Endpoint.URL	../TSUploadSupportingDocument	**Updated with the actual endpoint URL during the installation**
FileUpload.ConfirmationNumberService.Name	{http://xmlns.oracle.com/OTSS/ OTSSConfirmationIdService/ OTSSConfirmationIdBPELProcess} otssconfirmationidbpelprocess_client_ep	
FileUpload.ConfirmationNumberService.Port	OTSSConfirmationIdBPELProcess_pt	
FileUpload.ConfirmationNumberService.URL	../soa-infra/services/OTSS-PSRM/ OTSSConfirmationIdService/ otssconfirmationidbpelprocess_client_ep	
FileUpload.UCM.Service.Name	{urn:GenericSoap}GenericSoapService	
FileUpload.UCM.Port.Name	GenericSoapPort	
FileUpload.UCM.Endpoint.URL	../idcws/GenericSoapPort	**Updated with the actual UCM endpoint URL during the installation**
FileUpload.UCM.StatusCode.Success	0	**Definition of status code on successful upload**
FileUpload.UCM.User		**User ID for UCM connection; to be populated by the implementation**
FileUpload.UCM.DocumentType	Document	
FileUpload.UCM.SecurityGroup	Secure	
FileUpload.UCM.DocumentAuthor		**Default document author; to be populated by the implementation**
FileUpload.UCM.DocumentAccount	webUserId	**Xpath containing the value of the account of the document that will be**

Configuration Property	Default/Delivered Value	Description
		uploaded. Xpath is relative to head element
FileUploadSequence.Prefix	FUID	**Prefix for file upload confirmation ID**
FileUpload.DocumentLocatorNumberService.Name	{http://xmlns.oracle.com/OTSS/OTSSDocumentLocatorNumberService/ OTSSDocumentLocatorNumberBPELProcess} otssdocumentlocatornumberbpelprocess_client_ep	**Configuration for Document Locator Number service endpoint**
FileUpload.DocumentLocatorNumberService.Port	OTSSDocumentLocatorNumberBPELProcess_pt	
FileUpload.DocumentLocatorNumberService.URL	http://<server_name>:8051/soa-infra/services/ OTSS-PSRM/OTSSDocumentLocatorNumberService/ otssdocumentlocatornumberbpelprocess_client_ep	
FileUpload.DLN.ErrorCode	23000	**Error code for invalid document locator number**
FileUpload.Parameter.ErrorCode	23001	**Error code for missing file upload parameters**
FileUpload.Upload.ErrorCode	23002	**Error code for file upload error**
FileUpload.UCM.UploadFolderID	352885742199000201	**Default Content ID of the UCM file upload folder. This is used in case FileUpload.UCM.UploadFolderName is not specified. May be modified by the implementation**
FileUpload.UCM.UploadFolderName		**Document Upload Folder Name, to be populated by the implementation**
FileUpload.UCM.UploadFolderWebLocation		**Web folder location of the uploaded file. This is concatenated with the uploaded filename to generate the complete path to the file. To be populated by the implementation**
Service Name: **OTSSCreateEnrollmentServiceProvider**		
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
CreateEnrollment.Endpoint.URL	../TSEnrollmentServiceRequest	**Updated with the actual endpoint URL during the installation**
Service Name: **OTSSCreateEnrollmentEBF**		
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

Configuration Property	Default/Delivered Value	Description
		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
CreateEnrollment.Endpoint.URL	../TSEnrollmentServiceRequest	**Updated with the actual endpoint URL during the installation**
CreateEnrollmentSequence.Prefix	CEID	
CreateEnrollment.ConfirmationNumberService.Name	{http://xmlns.oracle.com/OTSS/Industry/Tax/ OTSSEnrollmentIDService} otssconfirmationidbpelprocess_client_ep	
CreateEnrollment.ConfirmationNumberService.Port	OTSSConfirmationIdBPELProcess_pt	
CreateEnrollment.ConfirmationNumberService.URL	../soa-infra/services/OTSS-PSRM/ OTSSConfirmationIdService/ otssconfirmationidbpelprocess_client_ep	**Updated with the actual endpoint URL during the installation**
CreateEnrollment.RequestMode.Sync	SYNCH	
CreateEnrollment.RequestMode.Async	ASYNCH	
FieldValue.Separator	:	
UserAcessUniqueConstraintViolation.ErrorNumber	30104	
MaxFailedEnrollmentAttempts.ErrorNumber	30105	
PendingEnrollmentFound.MessageNumber	30106	
MaxEnrolledUsersLimitReached.MessageNumber	30107	
CreateEnrollment.EnrollmentIDService.Name	{http://xmlns.oracle.com/OTSS/Industry/Tax/ OTSSEnrollmentIDService} OTSSEnrollmentIDGeneratorService	
CreateEnrollment. EnrollmentIDService.Port	OTSSEnrollmentIDService_pt	
CreateEnrollment. EnrollmentIDService.Endpoint.URL	http://<server_name>:8051/soa-infra/ services/ OTSS-PSRM/OTSSEnrollmentIDService/ OTSSEnrollmentIDGeneratorService	**Updated with the actual endpoint URL during the installation**
Service Name: **OTSSSummaryEnrollmentEBF**		
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
EnrollmentSummary.Service.Name	{http://ouaf.oracle.com/spl/XAIXapp/ xaiserver/ TSGetEnrollmentSummary } TSGetEnrollmentSummaryService	
EnrollmentSummary.Port.Name	TSGet EnrollmentSummaryPort	

Configuration Property	Default/Delivered Value	Description
EnrollmentSummary.Endpoint.URL	.../TSGetEnrollmentSummary	**Updated with the actual endpoint URL during the installation**
Service Name: **OTSSQueryEnrollmentEBF**		
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
EnrollmentInquiry.Service.Name	{http://ouaf.oracle.com/spl/XAIXapp/xaiserver/ TSGetUserEnrollment}TSGetUserEnrollmentService	
EnrollmentInquiry.Port.Name	TSGetUserEnrollmentPort	
EnrollmentInquiry.Endpoint.URL	.../ TSGetUserEnrollment	**Updated with the actual endpoint URL during the installation**
Service Name OTSSUserAccessService		
Max.EnrollmentAttempts	5	
Max.EnrolledUsers	<LineOfBusiness name="IND">2</LineOfBusiness> <LineOfBusiness name="BUS">2</LineOfBusiness>	
DefaultMax.EnrolledUsers	2	
CheckEnrolledUsersCount.Flag	true	
UserAccessService.Approval.Name	{http://xmlns.oracle.com/OTSS/Industry/Tax/ OTSSUserAccessApprovalService} OTSSUserAccessApprovalService	
UserAccessService.Approval.Port	OTSSUserAccessApprovalService_pt	
UserAccessService.Approval.URL	../soa-infra/services/OTSS-PSRM/OTSSUserAccessApprovalService/OTSSUserAccessApproval	**Updated with the actual endpoint URL during the installation**

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, right-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
- **CIID** – ContactInfoSequence
- **AMID** – AddressMaintenanceSequence
- **CEID** – CreateEnrollmentSequence
- **TFID** – TaxFormSequence
- **RFID** - RegistrationFormSequence
- **FUID** - FileUploadSequence

Also see Appendix B for a sample of the *GetConfirmationID Request/Response* messages.

Integration Services

Name	Description
OTSSEnrollmentIDService	Enrollment ID SOA composite. The process uses the synchronous flow returns a unique alphanumeric ID as output.

Adapter Services

Name	Description
OTSSConfirmationIdAdapter	Confirmation number SOA composite. OTSS confirmation number DB adapter.

Enrollment ID Utility Service

This service is used to generate a unique ID for the user enrollment process.

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 the *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 the *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
TSOneTimePayment	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 PAYMENT_REF_ID – A unique transaction ID.
    - Column PAYMENT_VENDOR – Payment vendor identifier in the revenue management system.
    - Column TAXPAYER_ID – The value of the **<taxpayerId>** element.
    - Column BEHALF_OF_TAXPAYER_ID – The value of the **<onBehalfOfTaxpayerId>** element.
    - Column PAYMENT_REF_STATUS – The value configured in the **ExternalpaymentData.Status.Pending** property (see *Module Configurations*).
    - Column ACCESS_TYPE_CD
    - Column KEY_NAME_1
    - Column KEY_VALUE_1

The following DVM-based translations are performed for both response and request messages:

- **<paymentVendor>** is translated using the *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 **TS_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
PAYMENT_VENDOR	Varchar 50	paymentVendor
PAYMENT_REF_ID	Varchar 50	externalId
TAXPAYER_ID	Varchar 50	taxpayerId
BEHALF_OF_TAXPAYER_ID	Varchar 50	onBehalfOfTaxpayerId
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*.
ACCESS_TYPE_CD	Varchar 50	accessTypeCd
KEY_NAME_1	Varchar 30	keyName1
KEY_VALUE_1	Varchar 30	keyValue1
KEY_NAME_2	Varchar 30	keyName2
KEY_VALUE_2	Varchar 30	keyValue2
KEY_NAME_3	Varchar 30	keyName3
KEY_VALUE_3	Varchar 30	keyValue3
KEY_NAME_4	Varchar 30	keyName4
KEY_VALUE_4	Varchar 30	keyValue4
KEY_NAME_5	Varchar 30	keyName5
KEY_VALUE_5	Varchar 30	keyValue5
KEY_NAME_6	Varchar 30	keyName6
KEY_VALUE_6	Varchar 30	keyValue6
KEY_NAME_7	Varchar 30	keyName7
KEY_VALUE_7	Varchar 30	keyValue7
KEY_NAME_8	Varchar 30	keyName8
KEY_VALUE_8	Varchar 30	keyValue8

Column Name	Datatype	Mapping
KEY_NAME_9	Varchar 30	keyName9
KEY_VALUE_9	Varchar 30	keyValue9
KEY_NAME_10	Varchar 30	keyName10
KEY_VALUE_10	Varchar 30	keyValue10

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.

Name	Description
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 the *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 the *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.

User Access Store

Column	Comments
USER_ACCESS_ID	Number 10. Unique identifier for the user access record.
USER_ID	Web user ID; matches the record in the OID.
ENROLLMENT_ID	Char 32. Unique ID used to track enrollment event.
LINE_OF_BUSINESS_CD	Indicates user's area of activities. Valid Values match values of LINEOFBUSINESS WSS Lookup.
USER_ROLE	Char 10. Level of user’s access to the account. Defaulted to OWNER
ACCESS_TYPE_CD	Char identifies the name of the access key set, e.g, TAXROLE.
REVENUE_MNGT_CD	Char 10. Identifies a back-end system where the record with access keys resides

Column	Comments
Access Keys 1 – 10: KEY_NAME_1 KEY_VALUE_1 KEY_NAME_2 KEY_VALUE_2 ……	These columns contain key names and values whose combination identifies a high-level object in the revenue management system to which the user has access. KEY_NAME_N – CHAR 30 KEY_VALUE_N – CHAR 30
STATUS	Indicates whether enrollment record is active Valid Values : • A – Approved • H – On Hold • B – Blocked • 1 .. N – Indicates the number of failed attempts
CREATED_BY	Web user id that creates the user access record
CREATION_DATE	Indicates the date when the user access record was created
LAST_UPDATED_BY	Web user ID that last updated the user access record
LAST_UPDATE_DATE	Keeps track of the date when the user access record was last updated

For additional details, see the *PSRMSS Intallation Guide*.

User Enrollment Request Integration Flow

Business Details

The self-service portal user selects an enrollment option associated with a line of business and provides set of identification details as an input.

The request is sent to the revenue management system. Based on the input, the revenue management determines which tax accounts owned by the user and retrieve the corresponding taxpayer and tax account identifiers (one or more keys) and the confirmation information.

Tax account identifiers are processed by the integration layer and captured in user access store, and the self-service user receives confirmation ID and details

Technical Details

This synchronous flow is one of the idiosyncratic *User Enrollment Flows*.

The self-service application sends the request in the form of an XML message to the integration.

The integration performs the following:

Confirmation ID is generated using *Confirmation Number Utility Service*

Previous Enrollment Attempts Verification invokes DB Adapter to retrieve an enrollment record for the input Line Of Business. If not found, the new Enrollment ID is generated using *Enrollment ID Utility Service*. If existing enrollment record found, check the status:

Blocked (B) - return error message back to the portal

Other than Blocked - copy enrollment ID to the main request, and forward the request to the revenue management system

Response Processing invokes *User Access Service*

Successful response - the *User Access Service* inserts the new records into user access store. It checks enrollment count for each access keys set. If too many users have access to a specific account, the entry is added in status On Hold (H) and the activation request is forwarded to the Worklist for revenue management agency approval.

Error response the *User Access Service* is checking errors count. If the empty enrollment record in Error status already exists, it checks if the maximum number of attempts is reached. If so, the enrollment record status is set to Blocked(B). If that's the first error attempt, the service creates empty enrollment record in status 1 (First Error)

The final response with confirmation message is returned to the portal.

The following fields are translated using DVM:

<lineOfBusiness> is translated using **OTSS_LineOfBusiness**

<accessType> is translated using **OTSS_AccessTypes**

<status> is translated using **OTSS_EnrollmentStatus**

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
OTSSCreateEnrollmentEBF	Enrollment Request enterprise business flow process. The process uses 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
TSEnrollmentServiceRequest	This web service retrieves the user account access keys collection and confirmation details for the input line of business and user identification information.

Get User Enrollment Integration Flow

Business Details

On the self-service portal, the taxpayer navigates to My Accounts page. This flow is invoked immediately and the request is sent to the integration layer/revenue management system in order to evaluate and refresh user's enrollment.

The revenue management system is expected to re-evaluate user's information, identify new tax accounts user should have access to, and deliver the new access keys to the integration layer.

The integration layer orchestrates the insertion of new user access records (if any received) and re-queries the enrollment line of businesses.

The final response to the self-service portal is expected to answer the following questions:

whether this user is enrolled already;

if so, into what lines of business

Technical Details

This synchronous flow is one of the idiosyncratic *User Enrollment Flows*.

The self-service application sends the request in the form of an XML message to the integration.

The integration invokes DB Adapter to retrieve a distinct list of enrollment ID-s of approved access records. Logically, it checks if the user is already enrolled successfully and gets the id-s of successful enrollment events.

If the query returned no records, the empty response is returned to the portal application

If the query returned at least one enrollmentID:

The DB Adapter is invoked to retrieve all enrollment records for each enrollment ID.

The result is copied to the main request and the request is forwarded to the revenue management system for further verification and refresh

The response from revenue management system may contain a list of new enrollment access keys. Integration is processing the response as follows:

Invokes *User Access Service* to insert new user access records.

If maximum number of enrolled users per account is reached, the enrollment records are placed on hold and the task is forwarded to the Worklist application for approval

Invokes DB Adapter to query the distinct list of Line Of Business record where user is enrolled

The final response delivered to the portal application contains the collection of Line Of Business(es) where the user is already enrolled.

The following fields are translated using DVM:

<lineOfBusiness> is translated using **OTSS_LineOfBusiness**

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
OTSSQueryEnrollmentEBF	Query/Refresh Enrolment 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
TSGetUserEnrollment	This web service retrieves the current status of user enrollment - a collection of Line Of Business (codes) where user is currently enrolled and new access keys collection

Enrollment Summary Integration Flow

Business Details

The self-service portal user is navigating to the Account Info portal page. The request is sent to the revenue management system and the response contains the list of tax accounts to which this user has an access and summary details

Technical Details

This synchronous flow is one of the idiosyncratic *User Enrollment Flows*.

The self-service application sends the request in the form of an XML message to the integration. The integration is orchestrating the subsequent process as follows:

Invokes Adapter Service *Select* User Access Keys to query the tax accounts to which user has an access

Forwards the request further to the revenue management system

Gets the response that contains the list of enrollment summary records. Each entry on the list includes:

Taxpayer name, line of business and a source system code

Access type code and up to ten key name-key value pairs with tax account identifiers

An indicator if the tax account should be picked up by the portal page as a default account

Enrollment summary title and details parameters

The following fields are translated using DVM:

<lineOfBusiness> is translated using **OTSS_LineOfBusiness**

<accessType> is translated using **OTSS_AccessTypes**

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
OTSSSummaryEnrollmentEBF	User Enrollment Summary enterprise business flow process. The process uses the synchronous flow without confirmation ID with database query. This process will have transformation using DVMs, pre- and post-extension services, and custom transformations.

Web Services

Name	Description
TSGetEnrollmentSummary	This web service retrieves the confirmation details using the input confirmation ID.

User Access Service

This service encapsulates user enrollment verification and user access maintenance logic.

This service inserts new access keys sets into user access store.

It also checks two aspects of the response from revenue management system:

Number of Failed Enrollment Attempts. In this scenario the input ***isError*** indicator is set to ***true.*** The logic retrieves an enrollment record using the input Enrollment ID and checks the status.

If the enrollment record is not found, it creates a new one and sets the status to **1 (First Error).**

If the enrollment record is found, it checks the status against the max number of attempts allowed. (See *Service Configurations for more details.)*

If the max number is reached, the existing enrollment record status is sent to **Blocked (B).** The system sends an un-blocking request to the Worklist using *User Access Approval Service*.

If the max number is not reached yet, it updates the status of the enrollment record to **N (Nth Error**).

Too Many Users Access the Same Account In this scenario the input ***isError*** indicator is set to ***false*** and the input contains a **collection of new user access keys sets** and the **Line of Business**. The logic counts enrolled users for the access key set/access type combination for each input enrollment record. If the max number allowed for input Line of Business (See *Service Configurations for more details)* is reached, the system updates the specific access record's status to **On Hold (H)** and sends an un-blocking request to the Worklist using *User Access Approval Service*.

The user access keys are deleted from the response.

Integration Services

Name	Description
OTSSUserAccessService	User Access SOA composite The process will use the synchronous flow which takes user access keys, user id, enrollment id, line of business, failed attempts count, and isError indicator as input and returns the updated user access keys and hasRecordsOnHold indicator as output.

User Access Approval Service

Business Details

This service handles the inclusion of blocked and held enrollment records as tasks to the worklist application. The approval or rejection of the specific task is also handled in this service

Technical Details

The service receives a single enrollment record (enrollment ID, status, access keys, access type and line of business) and creates the corresponding human task.

The approval workflow depends on the status of the input enrollment record:

On Hold(H) - approving the task will just update the status of the user access keys record to Approved (A). Rejecting will delete the user enrollment record and user can enroll on the same account again.

Blocked (B) - approving will update the status of the user access keys record to **1 (First Error)** Rejecting will just do nothing and enrollment status remains blocked.

Integration Services

Name	Description
OTSSUserAccessService	User Access SOA composite The process will use the synchronous flow which takes user access keys, user id, enrollment id, line of business, failed attempts count, and isError indicator as input and returns the updated user access keys and hasRecordsOnHold indicator as output.

Tax Account Alerts Integration Flow

Business Details

The taxpayer selects one of the tax accounts and navigating to the Account Info portal page. The request is sent to the revenue management system and the response contains list of alerts related to the tax account

Technical Details

This flow conforms to the *Synchronous Flow without Confirmation ID* pattern.

The self-service application sends the request in the form of an XML message to the integration. The request contains an account identifier (set of access keys and access type) and the line of business. The integration forwards the request further to the revenue management system and gets the response that contains a list of alerts, each entry includes:

Alert Type

List of Alert Parameters

The following fields are translated using DVM:

<alertType> is translated using **OTSS_AlertType**

<lineOfBusiness> is translated using **OTSS_LineOfBusiness**

<accessType> is translated using **OTSS_AccessType**

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
OTSSGetTaxAccountAlertsEBF	Tax Account Alerts 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
TSGetTaxAccountAlerts	This web service sends account identifiers (the access keys and access types) and retrieves the list of account alerts

Account Summary Integration Flow

Business Details

The taxpayer selects one of the tax accounts and navigating to the Account Info portal page. The request is sent to the revenue management system and the response contains the high-level details of the tax account

Technical Details

This flow conforms to the *Synchronous Flow without Confirmation ID* pattern.

The self-service application sends the request in the form of an XML message to the integration. The request contains an account identifier (set of access keys and access type) and the line of business. The integration forwards the request further to the revenue management system and gets the response that contains:

Summary title parameters

Summary details parameters,

The address associated with the account

The tax account's current balance.

The following fields are translated using DVM:

<country> is translated using **OTSS_Country**

<taxType> is translated using **OTSS_TaxType**

<lineOfBusiness> is translated using **OTSS_LineOfBusiness**

<accessType> is translated using **OTSS_AccessType**

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
OTSSGetTaxAccountSummaryEBF	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
TSGetTaxAccountSummary	This web service retrieves tax account summary for the input account identifiers (access keys and access type) and the line of business

Filing History Integration Flow

Business Details

The taxpayer is navigating to the Account Info portal page and the request for account payment history is sent to the revenue management system. The response contains the filing history, each entry on the list describes one filing period. The taxpayer can limit the result by start/end date range

Technical Details

This flow conforms to the *Synchronous Flow without Confirmation ID* pattern.

The self-service application sends the request in the form of an XML message to the integration. The request contains an account identifier (set of access keys and access type) and optional date range. The integration forwards the request further to the revenue management system and gets the response that contains account's filing history.

The following fields are translated using DVM:

<filingStatus> is translated using **OTSS_FilingStatus**

<taxType> is translated using **OTSS_taxType**

<accessType> is translated using **OTSS_AccessType**

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
OTSSGetFilingHistoryEBF	Get Filing History 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
TSGetFilingHistory	This web service sends account identifiers (the access keys and access types) and retrieves account's filing history.

Payment History Integration Flow

Business Details

The taxpayer is navigating to the Account Info portal page and the request for account payment history is sent to the revenue management system. The response contains the list of payments. The taxpayer can limit the result by start/end date range

Technical Details

This flow conforms to the *Synchronous Flow without Confirmation ID* pattern.

The self-service application sends the request in the form of an XML message to the integration. The request contains an account identifier (set of access keys and access type) and optional date range. The integration forwards the request further to the revenue management system and gets the response that contains a list of payments.

The following fields are translated using DVM:

<status> is translated using **OTSS_PaymentStatus**

<paymentType> is translated using **OTSS_PaymentType**

<accessType> is translated using **OTSS_AccessType**

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
OTSSGetPaymentHistoryEBF	Get Payment History 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
TSGetPaymentHistory	This web service sends account identifiers (the access keys and access types) and retrieves list of payments.

Taxpayer Summary Integration Flow

Business Details

The taxpayer is navigating to the Taxpayer Info portal page. The request is sent to the revenue management system and the response contains the essential info about this taxpayer and also primary contact details (applicable for businesses)

Technical Details

This flow conforms to the *Synchronous Flow without 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. The request includes taxpayer ID and other access keys, access keys and also line of business.

The response that contains taxpayer name, taxpayer type and summary title and detail parameters, and also primary contact's type, name and email address

The following fields are translated using DVM:

<lineOfBusiness> is translated using **OTSS_LineOfBusiness**

<taxpayerType> is translated using **OTSS_TaxpayerType**

<accessType> is translated using **OTSS_AccessType**

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
OTSSGetTaxpayerSummaryEBF	Get Taxpayer Summary 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
TSGetTaxpayerSummary	This web service retrieves the essential taxpayer details using input taxpayer ID, access keys, access type and line of business..

Taxpayer Contact Info Integration Flow

Business Details

The taxpayer is navigating to the Taxpayer Info portal page. The request is sent to the revenue management system and the response contains the list of addresses associated with this taxpayer.

Technical Details

This flow conforms to the *Synchronous Flow with Confirmation ID* pattern.

This flow operates in two modes, depends on the action (value of the **<action>** element):

Read (Action READ) - initially, 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 back the response containing the list of phone numbers and an email address associated with the input taxpayer (first access key).

Update (Action UPDATE) - The taxpayer may choose to modify the information and then the request is sent to the revenue management system. In this case the response contains confirmation details.

The following fields are translated using DVM:

<phoneType> is translated using **OTSS_PhoneType**

<accessType> is translated using **OTSS_AccessType**

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
OTSSGetTaxpayerContactInfoEBF	Get Taxpayer Contact Info 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
TSGetTaxpayerContactInfo	This web service retrieves the list of phone numbers and an email address associated the input taxpayer ID in the request header.

Taxpayer Correspondence Info Integration Flow

Business Details

The taxpayer is navigating to the Taxpayer Info portal page. The request is sent to the revenue management system and the response contains the list of addresses associated with this taxpayer.

Technical Details

This flow conforms to the *Synchronous Flow without 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 following fields are translated using DVM:

<country> is translated using **OTSS_Country**

<addressType> is translated using **OTSS_AddressType**

<accessType> is translated using **OTSS_AccessType**

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
OTSSGetTaxpayerCorrespondenceInfoEBF	Get Taxpayer Correspondence 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
TSGetTaxpayerCorrespondenceInfo	This web service retrieves the list of taxpayer's addresses using the input taxpayer ID in the request header.

Address Maintenance Integration Flow

Business Details

The taxpayer provides the address details and address change reason when applicable. The information is sent to the revenue management system and the taxpayer receives the confirmation of this transaction.

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 following fields are translated using DVM:

<country> is translated using **OTSS_Country**

<addressType> is translated using **OTSS_AddressType**

<addressChangeReason> is translated using **OTSS_AddressChangeReason**

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
OTSSAddressMaintenanceEBF	Address Maintenance enterprise business flow process. The process uses 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
TSAddressMaintenance	This web service is used to add or update the single address and returns confirmation ID.

Process Tax Form Integration Flow

Business Details

The taxpayer is filing the tax return or other tax-related form online. This is an interactive process, and the data entered may be validated and re-entered multiple times before final submission. User may request to copy the information from the previous return and/or re-calculate the numbers, or perform other custom actions. When form is finally submitted, the taxpayer receives Document Locator Number that can be used to track the form across the enterprise.

Technical Details

This combined synchronous/asynchronous flow is one of the idiosyncratic *Form Process Flows*.

The form data is sent from the portal application along with action code (value of the element **<action>**.

The flow continues depends on the action:

Validation logic is performed if the input action is VALIDATE or READY.

If the **<validationRule>** element is populated, the process retrieves validation service URL, populates validation rule name on the input and invokes *OTSSFormValidationService* (see below). The out-of-box solution includes validation service via Oracle Policy Automation rulebase deployed on Oracle Determination Server.

The response from validation service contains updated form data and exceptions collection. The data is copied back to the main request.

If the validation returned no errors the process continues, otherwise the response is returned back to the portal application.

The process determines if the revenue management system supports on-demand form data validation. It checks whether configuration property Backend.Validation.Enabled is set to **true**. If so, the request is forwarded to the revenue management system for validation.

The response is expected to contain updated form data and exceptions collection.

All other Actions, including Custom Actions logic is performed if the input action is anything but VALIDATE or READY or SUBMIT.

The request is sent to the revenue management system. The response is expected to contain updated form data and exceptions collection.

Submission logic is performed if the input action is SUBMIT.

The process determines endpoint URLs of confirmation ID and document locator number generation services and calls the Confirmation ID generation service, and then Document Location Number generation service (the latter with action specified in **DLN.Action.Create** property).

The following process is controlled by <responseMode> element on the input message that is populated based on the admin configuration in portal application. The valid values are: ASYNCH and SYNCH.

If the response mode is SYNCH then the process invokes the web service and forwards the message to the revenue management system.

If the response mode ASYNCH the transformed message is sent to the queue using JMS Producer adapter and the response containing confirmation ID and document locator is sent to the portal application.

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
OTSSProcessTaxFormEBF	Process Tax Form enterprise business flow process. The process uses the special combined synchronous/asynchronous flow with confirmation ID generation, document locator number generation and form data validation. This process will have transformation using DVMs, pre- and post-extension services, and custom transformations.

Adapter Services

Name	Description
OTSSProcessTaxFormJMSProducer	Self-service application Process Tax Form JMS producer. JMS producer adapter to write to the self-service application tax form queue.
OTSSProcessTaxFormtJMSConsumer	OTSS Process Tax Form JMS consumer. JMS consumer adapter to read from the OTSS service tax form queue.

JMS Queues

Name	Description
OTSSProcessTaxForm	Self-service application Process Tax Form queue. Used by the integration layer to add transformed tax form messages from the self-service application.
OTSSProcessTaxError	Self-service application Process Tax Form error queue. Error Queue for self-service application Process Tax Form.

Web Services

Name	Description
TSProcessTaxForm	This web service handles all aspects of online form processing: intermediate form data validation, custom actions, and final form submission. Depending on the action, the response contains either an updated form data and the list of exceptions or the confirmation ID and details.

Process Registration Form Integration Flow

Business Details

Similar to Process Tax Form Integration flow, *Business Details*.

Technical Details

Similar to Process Tax Form Integration flow, *Technical Details* .

Integration Services

Name	Description
OTSSProcessTaxFormEBF	Process Tax Form enterprise business flow process. The process uses the special combined synchronous/asynchronous flow with confirmation ID generation, document locator number generation and form data validation. This process will have transformation using DVMs, pre- and post-extension services, and custom transformations.

Adapter Services

Name	Description
OTSSProcessRegistrationFormJMSProducer	Self-service application Process Tax Form JMS producer. JMS producer adapter to write to the self-service application tax form queue.
OTSSProcessRegistrationFormtJMSConsumer	OTSS Process Tax Form JMS consumer. JMS consumer adapter to read from the OTSS service tax form queue.

JMS Queues

Name	Description
OTSSProcessRegistrationForm	Self-service application Process Tax Form queue. Used by the integration layer to add transformed tax form messages from the self-service application.
OTSSProcessRegistrationFormError	Self-service application Process Tax Form error queue. Error Queue for self-service application Process Tax Form.

Web Services

Name	Description
TSProcessRegistrationForm	This web service handles all aspects of online form processing: intermediate form data validation, custom actions and final form submission. Depends on the action the response contains either an updated form data and the list of exceptions or the confirmation ID and details.

Form Validation Integration Flow

Business Details

The online form data needs to be validated and the validation rule is specified on form definition (self service portal application configuration). Integration layer detects validation rule name and invokes this service to perform the validation.

The response from this service contains an updated form data and a collection of exceptions.

Technical Details

This flow is synchronous pattern and it communicates with Oracle Determinations Server via web service. This integration is using generic WSDL; the endpoint URL is calculated dynamically by concatenation of the Oracle Determinations Server location (flow's input), web services directory and operation name (service configurations) and validation rule name (flow's input) (see *Service Configurations* for more details).

The form data is transformed into rulebase web service format and back via XSL. The transformation is based on the assumption that the rulebase was created using OPA Data Model Generator in portal self-service application. It relays on a specific data model structure and element's public names generation rules.

For each form line the transformed rulebase-bound request contains an original value sent from the portal application.

The response from the rulebase contains determined values for calculated or corrected lines and/or an error message. The transformation applies the updates and creates updated form data. form data The errors received from the rulebase are collected and added to the exceptions list.

The response from the flow contains updated form data and the exceptions collection.

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
OTSSFormValidationService	Refresh Lookup enterprise business flow process. The process uses the synchronous flow..

Web Services

Name	Description
OPA Rulebase web service, generic WSDL	When this web service is invoked with operation **Assess**, it receives the form data and returns the validated form data and exceptions list. One web service per rulebase is available when the rulebase is deployed on Oracle Determinations Server .

Upload Supporting Document Integration Flow

Business Details

The taxpayer uploads supporting documents (scanned receipts, letters or other) immediately following the online form filing or later, upon request from revenue management agency. The documents need to be associated with either a specific form or other self-service transaction.

Taxpayer enters document locator or reference transaction confirmation ID and uploads file from the local file system. The request is sent to the integration layer. This flow orchestrates file delivery to a document repository, generates confirmation ID for the transaction and sends file location, document locator and confirmation ID to the revenue management system so it may establish the link between the form and the uploaded file. The response from the revenue management system contains confirmation information.

Technical Details

This synchronous flow is one of the idiosyncratic *Form Process Flows*.

The self-service application sends the request in the form of an XML message to the integration. The request contains: uploaded file contents as base64-encoded stream, the original file name (with extension) and either document locator number or reference confirmation ID.

The integration flow performs the follows:

Document Locator Number Validation. If the input request contains document locator number, the system checks its validity. The validation service endpoint URL is calculated based on service configurations. Out-of-box solution invokes OTSSDocumentLocatorNumberBPELProcess.

Document Name Assignment the original document name is replaced with concatenation of reference ID (either document locator number or reference confirmation ID) concatenated with current timestamp, to ensure its uniqueness. For example, if the input document locator number is TFDLN00000005082, the file name may become TFDLN00000005082-1396561401300.

Document Upload Service Call the details of the target document repository are calculated and the document contents are sent there with synchronous web service request. The response from the document repository expected to contain file location. The out-of-box solution includes interaction with UCM.

Confirmation ID is generated upon successful response from the document repository.

File Location is forwarded to the revenue management system along with the original input document location number (or reference confirmation ID) and the newly-generated confirmation ID for the upload transaction.

File Contents are sent to the back-end system if the **RetainContent** configuration property is set to ***true.***

See *Service Configurations* for more details.

The final response from the revenue management system 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
OTSSFormUploadEBF	Upload Document enterprise business flow process. The process uses the special synchronous flow with confirmation ID. This process will have transformations, confirmation ID generation, document locator number validation, pre- and post-extension services, and custom transformations.

Web Services

Name	Description
TSUploadSupportingDocument	This web service uploads file contents to the document repository and receives back the confirmation details.

Print Form Document Integration Flow

Business Details

The user has the opportunity to print or save the form upon submission. When **Print** button pressed, the request form data is sent to the integration layer and this flow is triggered. The printable document file is created and sent back to the portal application as base64-encoded byte stream. The response also contains the generated document's MIME type.

The actual document generation is performed by an external system. Out-of-box solution includes web service-based interaction with Oracle Business Intelligence Publisher.

Technical Details

This synchronous flow is one of the idiosyncratic *Form Process Flows*.

The self-service application sends the request to the integration. The request contains a document data in form of a valid XML.

The request is transformed as follows based on the service configurations (see *Service Configurations* for more details).

Data Encoding. There is an encoding indicator. If set to **true**, the document data XML is base64 encoded. Encoding Service endpoint URL is determined based on the configurations. Base solution includes OTSSBase64EncodedDataGeneratorService.

Data Transformation additional custom transformation is applied.

Document Generation Service the dynamic endpoint is calculated based on the configuration.

BI Publisher-specific details the out-of-box solution uses Oracle BI Publisher for document generation. The specific inputs for Oracle BI Publisher's ReportService include: report name and location, user and password for the connection and the output document's format. All these are defined in *Service Configurations* and can be customized by the implementation.

After completing the transformations the process invokes Document Generation Service web service. The response is transformed and returned to the self-service portal.

If document generation is failed, the final response is populated with pre-configured error message number. 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
OTSSPrintFormEBF	Print Form enterprise business flow process. The process uses the special synchronous flow. This process will have transformation using DVMs, pre- and post-extension services, and custom transformations.

Web Services

Name	Description
TSPrintForm	This web service retrieves the document contents base64-encoded and MIME document type.

Document Locator Number Utility Service

Process Tax Form, Process registration Form and Upload Supporting Document integration processes use a unique document locator number on the request XML message. This unique identifier is communicated to all integration participants and can be used for form tracking purposes across the enterprise.

The Document Locator Number Utility Service encapsulates the common document locator number generation and validation logic. The integration flows which require document locator number generation or validation invoke this service using dynamic partnerlink URL.

The input to this service is an alphanumeric prefix for the document locator name; the output is the alphanumeric document locator number, calculated and formatted as follows:

The **final length** of the output document locator is derived from *Module Configurations*.

The size of the numeric portion **N** is calculated as **final length -1 - prefix' size**. One position is reserved for the check digit.

The next available sequence number derived.

The check sum is generated for the sequence number based on Luhn's algorithm and appended to the end of the sequence number.

Note:

The Luhn algorithm or Luhn formula, also known as the "modulus 10" or "mod 10" algorithm, is a simple checksum formula used to validate a variety of identification numbers, such as credit card numbers, IMEI numbers, National Provider Identifier numbers in US and Canadian Social Insurance Numbers. The algorithm is in the public domain and is in wide use today. It is specified in ISO/IEC 7812-1.[1] It is not intended to be a cryptographically secure hash function; it was designed to protect against accidental errors, not malicious attacks. Most credit cards and many government identification numbers use the algorithm as a simple method of distinguishing valid numbers from collections of random digits.

The result is formatted as **N**-digit number, right-padded with zeroes.

The prefix is concatenated with N-digit number and returned to the caller process.

The delivered integration configuration includes a prefix for each flow.

The service uses a DB Adapter to get the next sequence number.

Also see *Appendix B* for a sample of the *Get Document Locator Number Request/Response* messages.

Integration Services

Name	Description
OTSSDocumentLocatorNumberBPELProcess	Document locator number SOA composite. The process uses the synchronous flow which takes a document locator prefix and the action as input and returns: • Action **CREATE** - generated document locator • Action **VALIDATE** - boolean indicator (true/false)

Adapter Services

Name	Description
OTSSDocumentLocatorNumberAdapter	OTSS Document Locator Number DB adapter.

Get Active Form Types Integration Flow

Business Details

The taxpayer provides the form category as an input. The language currently used by the taxpayer is also populated on the request. The request is sent to the revenue management system and the response contains the list of form types that are eligible for import.

Technical Details

This flow conforms to the *Synchronous Flow without 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 form types list.

The following elements are transformed using DVM:

<language> element value is translated using *OTSS_Language*.

<formSubType> element value is translated using *OTSS_FormCategory*.

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
OTSSRetrieveActiveFormTypesEBF	Retrieve Active Form Types 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
TSRetrieveActiveFormTypes	This web service retrieves the list of available form types with descriptions for the given form category and language

Import Form Definitions Integration Flow

Business Details

The taxpayer selects one or more form types as an input. The request is sent to the revenue management system and the response contains the definitions for each form type and a collection of lookups. Lookup represents a list of valid values for the given form line. Textual information is retrieved in multiple languages supported by revenue management system.

Technical Details

This flow conforms to the *Synchronous Flow without 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 the requested form types and lookup collection.

The following elements are transformed using DVM:

- **<language>** element value is translated using *OTSS_Language*.
- **<occurrence>** element under **<section>** group is translated using *OTSS_FormSectionOccurence*.
- **<dataType>** element under **<field>** group is translated using *OTSS_FormDataType*.

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
OTSSRetrieveFormTypeDefinitionsEBF	Retrieve Form Definitions 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
TSRetrieveFormTypeDefinitions	This web service retrieves the definitions of the input form types. The response also contains a collection of Lookups

Refresh Lookup Integration Flow

Business Details

The taxpayer selects a Lookup that was initially imported from the revenue management system and requests to update the lookup values.. The request is sent to the revenue management system and the response contains the current list of lookup values with descriptions in all languages supported by revenue management system.

Technical Details

This flow conforms to the *Synchronous Flow without 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
OTSSRfreshFormLookupEBF	Refresh Lookup enterprise business flow process. The process uses the synchronous flow without confirmation ID. This process supports pre- and post-extension services, and custom transformations.

Web Services

Name	Description
TSRefreshFormLookup	This web service retrieves the lookup values with descriptions in all languages supported by revenue management system.

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 TS_PaymentDestination DVM.

- <paymentType> is translated using the *OPC_PaymentType DVM*. The value represents source of the payment: checking or savings bank account, credit card or other. Official Payments Corporation 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 exists, 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 PSRM 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.

Name	Description
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.

Name	Description
	SOA composite uses the JMS adapter to read the message from the Payment Report queue and invokes the PSRM 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.
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
TSProcessExtPayReportRecord	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-PSRM. All composite processes deployed for integration are available under the partition OTSS-PSRM.
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

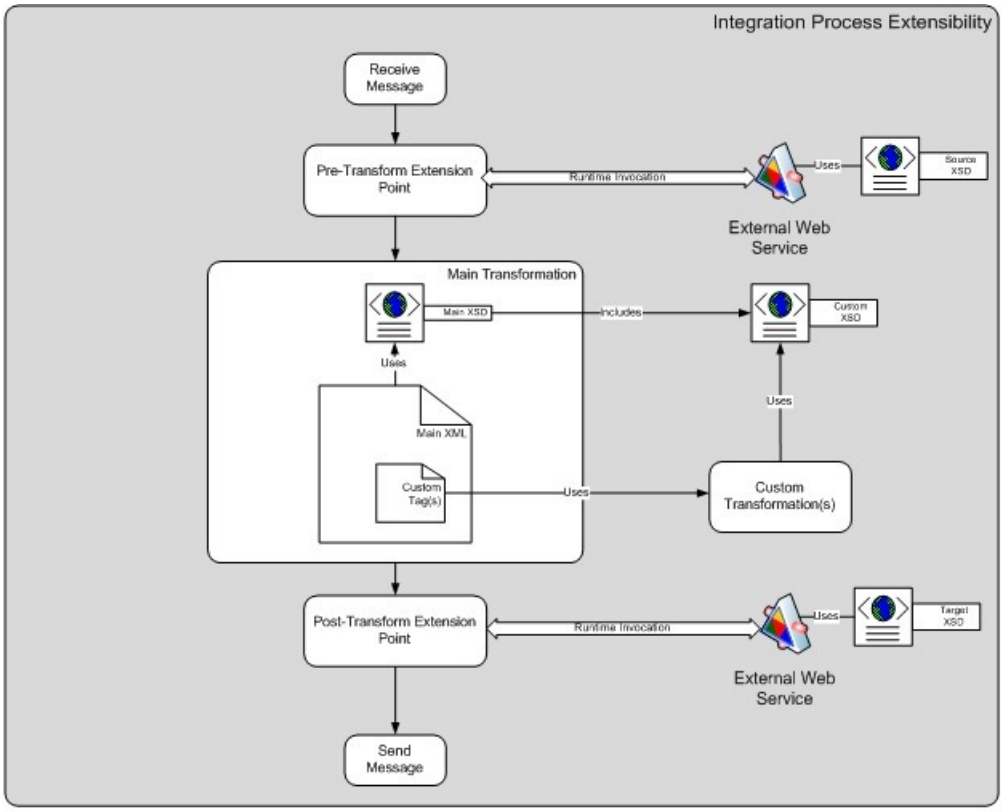
- In the WebLogic console, navigate to **Services** > **Messaging** > **JMS Modules** .
- Select the OTSS-PSRM Integration JMS Module to display all queues related to this integration.
- Select the appropriate error queue and click the Monitoring tab. This tab displays the details about messages in the 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. All processes except the asynchronous Taxpayer Service Request	Business Error	Error is sent to the Web Self Service application	None	Retry can be triggered by the self service user
	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

Integration Flow	Type of Error	Action	Notification Type	Retry
				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-PSRM/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 the **DeployComposite** action. See the *PSRMSS Installation Guide* for additional details on this process.

- **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-PSRM/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 *PSRMSS Installation Guide*.

  After deploying the files to MDS, restart the SOA server.

Document Locator Number Utility

The confirmation number assignment can be customized for all or some of the integration flows.

- **Option 1**. Modify the result using configurations only.

  Locate the size property **DLN.Size** (see the *Module Configurations* topic for details) and the prefix property **TaxForm.DLN.Prefix** (see *Service Configurations*) in the `ConfigurationProperties.xml` file and change the values.

  > **Note:** The maximum length of the Document Locator Number, including the prefix, should not exceed 50 characters.

  Modify the `$PRODUCT_HOME/ /services/industry/Tax/EBF/OTSSDocumentLocatorNumberService/xsl/Xform_GetDocumentLocatorNumber.xsl` file.

  Update the prefix check with the new value.

  Redeploy the **OTSSDocumentLocatorNumberService** process using the **DeployComposite** action. See the *PSRMSS Installation Guide* for additional details on this process.

- **Option 2**. Replace the confirmation number generation logic.

  Create your own document locator number generation service(s) using the integration **OTSSDocumentLocatorNumberBPELProcess** WSDL located at:

  `$PRODUCT_HOME/MDS-Artifacts/OTSS-PSRM/AIAMetaData/AIAComponents/BusinessProcessServiceLibrary/OTSSDocumentLocatorNumberService/`

  Deploy this service at the location accessible from the SOA server.

In the **ConfigurationProperties.xml** file, update the service-level properties for those services that should be using a new document locator number utility service:

- `[service name].DLNService.Name`
- `[service name].DLNService.Port`
- `[service name].DLNService.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 *PSRMSS 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-PSRM/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 WSDLs in product install home are the following:

```
$PRODUCT_HOME/MDS-Artifacts/OTSS-PSRM
```

```
/AIAMetaData/AIAComponents/ExtensionServiceLibrary/
```

Then deploy the concrete WSDLs 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://<server_name>:8001/soa-infra/services/Extension/
OTSSPaymentExtensionProcess/OTSSPaymentExtensionService"/>-->
<soap:address location="<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:

```
OTSSPSRM/MDS-Artifacts/OTSS-PSRM/AIAMetaData/AIAComponents/ ApplicationObjectLibrary/OTSS/
V1/schemas
OTSSPSRM/MDS-Artifacts/OTSS-PSRM/AIAMetaData/AIAComponents/
ApplicationObjectLibrary/OPC/V1/schemas
...etc.
```

The custom XSL files are located in product install home under the directory:

```
OTSS-PSRM/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 14

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, refer to *Oracle WebCenter* documentation. For more detailed information on WebCenter task flows and more advanced customization techniques, see the *Oracle Fusion Middleware Developer's Guide for Oracle WebCenter 11g*.

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, see the *Oracle WebCenter* documentation.

Portal Navigation

The base product implements portal navigation by referencing a navigation model. Navigation model provides out-of-the-box multi-language support and allows product to utilize WebCenter's own task flows for left-side navigation menu and the breadcrumbs navigation.

Navigation models are managed on the WebCenter Administration Console and can be added/maintained/customized via WebCenter Composer. Refer to Application Bookmarks chapter for important customization remarks.

Navigation Components

The base product is provided with a single default navigation model, **WSSPortalNavigation**, which uses three portal navigation components:

- The **top menu bar** is part of the base page template.
- The **breadcrumbs navigation taskflow** is a part of the base page template. It displays the "path" that user has followed starting from the top-level menu entry. The task flow included in the base page template has the following parameters:
  - **Navigation** [/oracle/webcenter/portalapp/navigations/WSSPortalNavigation.xml]. This parameter references the full path to an actual navigation model source file. If your implementation wishes to use the custom navigation model, this parameter should be modified accordingly.
  - **Start path** [/]. This value indicates that the breadcrumbs must be shown starting from the highest possible level of the hierarchy.
  - **Show Root** [false]. This value indicates that the navigation model's root node should not be shown.
  - Display style [Horizontal]. This parameter controls the direction of the crumbs.
- The **left-side navigation panel** is a task flow included in WebCenter's default resource catalog. It displays the navigation model in as an expandable tree. Most base product pages include this component with the parameters set as follows:
  - **Navigation** [/oracle/webcenter/portalapp/navigations/WSSPortalNavigation.xml]. This parameter references the full path to an actual navigation model source file. If your implementation wishes to use a custom navigation model, this parameter should be modified accordingly.
  - **Start path** #{navigationContext.navigationModel['modelPath=/oracle/webcenter/portalapp/navigations/WSSPortalNavigation.xml'].currentSelection.prettyUrlPath[1]}. This parameter defines the starting point of the navigation. Note that this expression returns a current page, not a sub-page.

- **Show levels** [3]. This parameter controls how many levels of the navigation model should be displayed on the tree.
- **Show Root** [#{box}]. This parameter indicates that the starting node should be displayed in the box.
- **Style** [#{collapsed}]. This parameter controls how the tree is initially displayed.
- **Show icons** [false].

Application Bookmarks

Navigation in the portal application is typically done by either using the navigation menus (the top bar or the side panel) or using various hyperlinks provided by the portal application itself. For example, such hyperlink is used for navigation to My Accounts page in order to switch the tax account in context.

The navigation model is customizable. Implementers may wish to change the location of the existing portal pages in the overall navigation hierarchy. For example, you may choose to move the **My Accounts** page under the **Online Services** page group.

To ensure the integrity of the internal application navigation when modifying the navigation model, the application defines a set of Application Bookmarks. Each bookmark is referencing the location of a base product page. In a runtime the application derives the navigation target from the bookmark rather than point directly to a page location.

Application Bookmarks are defined as a value of a Lookup **SYS_BOOKMARKS**.

This lookup includes the values of all the portal pages. For each defined value the Extended Value captures the URL of the page according to the current navigation model. For example, the navigation URL for Service Requests is `/faces/service/servicereq`.

- All URLs have the "/faces" prefix.
- The "/service" represent the Online Services page and the value "service" is the ID of the Navigation Item of that page in the navigation model.
- The "/servicereq" represent the Service Request page and the value "servicereq" is the ID of the Navigation Item of that page in the navigation model.

Changing Bookmarks

If the page hierarchy of the current navigation model is modified, changes should propagated to the application bookmarks. For example, if you move the **My Accounts** page under the **Online Services** page group, its navigation URL (in the application bookmarks definition) should change from `/faces/myaccount` to `/faces/service/myaccount`.

To amend the application bookmarks, a new set of bookmarks must be created and noted in the system configuration options, following these steps:

- Duplicate the SYS_BOOKMARKS lookup using the lookup **Duplicate** button and manipulate the new lookup's values. For example, change the Extended Value of the bookmark MY_ACCOUNT (lookup value **MY_ACCOUNT**) from `/faces/myaccount` to `/faces/service/myaccount`.
- Update the system option DEF_SYS_BOOKMARKS to reference a newly created lookup.
- The portal application has to be restarted in order for the new system option to take effect.

Customizing Navigation

Various customization requirements are supported:

- **Modify the text of the navigation links.** The text of the navigation links can be modified via WebCenter Composer. Login as an administrator, select the navigation model from the portal resources list and edit the navigation items.
- **Modify/rearrange the existing navigation structure.** The navigation model is customizable. The items could be re-arranged, re-named and re-structured according to the business requirements.

  Note: Application Bookmarks must be updated if you modify the hierarchy of the pages provided with the base product, Please see Application Bookmarks chapter for more details.

- **Add new pages added to the portal.** When new page is added to the portal application, the corresponding entry should be added to the navigation model.

  Note: The top-level pages appear on the top menu bar. The pages added to the hierarchy beneath one of the top menu pages appear on the pull-down menus.

- **Create a special set of navigation links.** Your implementation may wish to support a scenario where one or more pages display a different set of navigation links. To create the links:
  - Create new navigation model.
  - Change the parameters of the left-side navigation task flow to reference the navigation model created above.
  - If the requirement is to override the navigation links on the top menu bar, you may create a new page template that is referencing the custom navigation model.

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 base product supports direct internal navigation to the portal pages and in some cases allows pre-loading a specific item. For example, the HTML may include a hyperlink ***Apply for Tax Clearance Certificate*** that initiates a specific service request. The hyperlinks can be placed in the application messages, summary pages of the interactive tax assistance, and/or everywhere else HTML is supported. The following table describes the syntax:.

Target	URL
Online Service	/psrmss/faces/service
Service Request – initiates specific service request	/psrmss/faces/service/servicereq?serviceReq=<Service Request Code>

Target	URL
Make a Payment – initiates specific payment with pre-populated payment amount (optional)	/psrmss/faces/service/payment?payDest=<Payment Destination Code>&payAmount=<Value>
Where Is My Refund? - initiates specific refund request	/psrmss/faces/service/refundstatus?refundReq=<Refund Request Code>
Track Your Transaction - initiates transaction tracking with confirmation ID pre-populated	/psrmss/faces/service/querytrans?CNFID=<Confirmation ID>
Tax Assistant - initiates specific interactive tax assistance	/psrmss/faces/taxassist?interviewSetCd=<Interview Set Code>&interviewCd=<Interview Code>
Tax Forms [List]	/psrmss/faces/service/fileonline/taxforms
Form [Specific Form] - initiates filing of a specific form	/psrmss/faces/oracle/webcenter/portalapp/pages/Form.jspx?FRM=<Form Type Code>
Upload File	/psrmss/faces/service/fileonline/uploaddoc
Business Registration [List]	/psrmss/faces/service/fileonline/regforms
My Accounts	/psrmss/faces/myaccount
Taxpayer Info	/psrmss/faces/taxpayerinfo
Account Info	/psrmss/faces/ myaccount/accountinfo
Settings	/psrmss/faces/sysadmin
System Options	/psrmss/faces/sysadmin/sysoption
Language	/psrmss/faces/sysadmin/syslang
Message	/psrmss/faces/sysadmin/sysmsg
Lookup	/psrmss/faces/sysadmin/syslookup
Field	/psrmss/faces/sysadmin/sysfield
Validation Rule	/psrmss/faces/sysadmin/sysval
Property Type	/psrmss/faces/sysadmin/sysprop
Email Definition	/psrmss/faces/sysadmin/sysemail
Alert Type	/psrmss/faces/sysadmin/sysalerttype
Configurations	/psrmss/faces/svcadmin
Service Request	/psrmss/faces/svcadmin/svcreq
Interview Set	/psrmss/faces/svcadmin/intrvset
Payment Destination	/psrmss/faces/svcadmin/paydest
Payment Provider	/psrmss/faces/svcadmin/paypvdr
Form Designer	/psrmss/faces/svcadmin/frmdsgnr?FRM=<Form Type Code>
Import Form Definition	/psrmss/faces/svcadmin/importfrm
Release Forms	/psrmss/faces/svcadmin/relfrm
Process Control	/psrmss/faces/svcadmin/procctrl
Data Type	/psrmss/faces/svcadmin/datatype
Access Type	/psrmss/faces/svcadmin/accesstype
Address Configuration	/psrmss/faces/svcadmin/addressconfig

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 "/psrmss" should be added to the link.

For example, the ITA page link will be:

```
/psrmss/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 `/psrmss/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 the implementation.

All images used in the portal application are included in 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 the 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 *PSRMSS 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 from the WebCenter administration console.

- Use the **Create Page** action to add a new page. The new page can be based on the base product page template, or any new page template created by the implementation.
- Use the **Edit Page** action to define the page structure. You may wish to add left-side navigation. The corresponding taskflow is available from the resource catalog. See *Portal Navigation* for the detailed description of navigation task flows parameters.
- Add a link to a new page to the navigation model used in your portal.

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:

```
/psrmss/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 Portal Page Content

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).

Most portal pages are provided with the left-side navigation panel. It may be removed or made invisible via WebCenter Composer, using the **Edit Page** action.

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 located 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 the 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.

The "breadcrumbs" navigation bar situated right below the page header may be removed or made invisible using the WebCenter Composer.

Note that changes made to a Page Template have global implications. They affect all the pages referencing the modified template.

> **Important:** If more advanced customizations, beyond what is described in this chapter, are required, it is recommended that the base product page template be copied and changes 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 the Portal Skin

The base product includes a default skin **WSSPortalTemplate_v1** (a .css file). UI elements reference the style selectors (classes, properties) and other definitions from the skin.

WSSRightOuterLayout	Alias associated with right hand layout of the page
WSSPageTitle	Alias associated with the page title.
WSSHelpImage	Alias associated with the help image.
WSSSubHeaderLayout	Alias associated with the subheader layout.
WSSSubHeader	Alias associated with the subheader.
WSSCommonIcon	Alias associated the images in the footer bar area.
WSSSiteBackgroundColor	Alias associated with the background color of the self-service application.

The skin is maintained using WebCenter Composer, and may be updated visually or by directly accessing the skin source code.

> **Important:** If skin changes are required, it is recommended that the base product skin be copied and changes 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

Process Tax Form (TSProcessTaxForm)

These are the node elements of the Processs Tax Form WSDL.

Node	Description
< TSProcessTaxForm>	This is the main node of the Process Tax Form WSDL. Its dateTimeTagFormat property is set to 'xsd'.
<head>	This node is a sub-node of <TSProcessTaxForm>. It contains the access keys and audit information of the request. Refer to *Access Keys and Audit Node*.
<lineOfBusiness>	This node is a sub-node of <TSProcessTaxForm>. It contains the line of business of the request.
<requestStatus> **<errorMessage>**	These nodes are sub-nodes of <TSProcessTaxForm>. They hold the details of the error message thrown by the web service. Refer to *Error Message Node*.
<confirmationData>	This node is a sub-node of <TSProcessTaxForm>. It holds the details of the confirmation message returned by the web service. Refer to *Confirmation Message Node*.
<linkedRequest>	This node is a sub-node of <TSProcessTaxForm>. It may contain the reference number identifying Taxpayer Identification performed for this form processing
<customInfo>	This node is a sub-node of < TSProcessTaxForm>. It holds the holds the details of any customization incorporated into the web service. Refer to *Custom Information Node*.
<mainData>	This node is a sub-node of <TSProcessTaxForm>. It contains the data for the request.
<validationRule/> **<validationServer/>**	These nodes are sub-nodes of < mainData> that are used to determine where the validation is invoked.

Node	Description
<formType/> **<formCategory/>** **<formData/>**	These nodes are sub-nodes of <mainData> that holds the form data. It contains the following information: formType - Contains the form type code in the revenue management system formCategory - Contains the form category formData - Contains the form content (structure and data)
<customProperties> **<sequence/>** **<propertyName/>** **<value/>** **</customProperties>**	This node is a sub-node of <mainData>. It contains optional form attributes that can be used for various purposes or capture an idiosyncratic information about forms of this type. sequence - Contains the order of the property propertyName - Contains the given name for the property value - Contains the value of the property
<exceptions> **<exception>** **<sequence/>** **<elementReference/>** **<errorMessage>** **<messageCategory/>** **<messageNumber/>** **<messageParameters>** **<parameters>** **<sequence/>** **<parameterType/>** **<parameterValue/>** **</parameters>** **</messageParameters>** **<currency/>** **<messageTxtOvrd/>** **<messageReference/> </errorMessage>** **</exception>** **</exceptions>**	This node is a sub-node of <mainData>. It contains the form exceptions encountered when validating the form. It holds the following: sequence - Contains the order of form exceptions elementReference - Contains the category for which the error originated messageCategory - Contains the message category number messageNumber - Contains the message number currency - Contains the currency code to be used for amount messageTxtOvrd - Contains the message text to display instead of the portal message messageReference - Contains details on the origin of message Holds parameters for the messages sequence - Contains the order of parameters parameterType - Contains the type of parameter parameterValue - Contains the value

Upload Supporting Document (TSUploadSupportingDocument)

These are the node elements of the Upload Supporting Document WSDL.

Node	Description
<TSUploadSupportingDocument>	This is the main node of the Upload Supporting Document WSDL. Its dateTimeTagFormat property is set to 'xsd'.
<head>	This node is a sub-node of <TSUploadSupportingDocument>. 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 <TSUploadSupportingDocument>. They hold the details of the error message thrown by the web service. Refer to *Error Message Node*.
<confirmationData>	This node is a sub-node of <TSUploadSupportingDocument>. It holds the details of the confirmation message returned by the web service. Refer to *Confirmation Message Node*.

Node	Description
<customInfo>	This node is a sub-node of <TSUploadSupportingDocument>. It holds the holds the details of any customization incorporated into the web service. Refer to *Custom Information Node*.
<mainData>	This node is a sub-node of <TSUploadSupportingDocument>. It contains the data for the request.
<documentLocator/> **<targetConfirmationId/>** **<fileURL/>** **<content/>**	These nodes are sub-nodes of <mainData>. It holds the document content in binary form to be stored in the document repository, the location of the stored document and document locator and/or target confirmation id to determine the the Form Service Task to which the file is related. documentLocator - Contains the document locator number of the tax form targetConfirmationId - Contains the form submission's original confirmation id fileURL - Contains document location URL generated by the document repository content - Contains the content of document in binary form

GetTax Account Summary (TSGetTaxAccountSummary)

These are the node elements of the Get Tax Account Summary WSDL.

Node	Description
<TSGetTaxAccountSummary>	This is the main node of the Get Tax Account Summary WSDL. Its dateTimeTagFormat property is set to 'xsd'.
<head>	This node is a sub-node of <TSGetTaxAccountSummary>. It contains the access keys and audit information of the request. Refer to *Access Keys and Audit Node*.
<lineOfBusiness>	This node is a sub-node of <TSGetTaxAccountSummary>. It contains the line of business of the request.
<requestStatus> **<errorMessage>**	These nodes are sub-nodes of <TSGetTaxAccountSummary>. They hold the details of the error message thrown by the web service. Refer to *Error Message Node*.
<customInfo>	This node is a sub-node of <TSGetTaxAccountSummary>. It holds the holds the details of any customization incorporated into the web service. Refer to *Custom Information Node*.
<responseData>	This node is a sub-node of <TSGetTaxAccountSummary>. It contains the response of the request.
<taxpayerName/> **<currentBalance/>** **<currency/>** **<taxType/>**	These nodes are sub-nodes of < responseData>. taxpayerName - Contains the name of taxpayer currentBalance - Contains the balance with P&I forecasting taxType - Contains the tax type
<summaryTitleParameters> **<sequence >** **<value/>** **</summaryTitleParameters>** **<summaryTextParameters>** **<sequence >** **<value/>** **</summaryTextParameters>**	This node is a sub-node of <responseData>. It holds the parameters for the title and detail messages configured in the portal. sequence - Contains the order of the parameter value - Contains the value of the parameter It may also contain the override messages to replace the portal configuration. overrideSummaryTitle - Contains override message for title override SummaryText - Contains override message for text

Node	Description
<overrideSummaryTitle/>	
<overrideSummaryText/>	
<location>	This node is a sub-node of <requestData>. It contains the address to be added or updated for the taxpayer.
<addressId/>	addressId - Uniquely identifies the address in the revenue management system
<addressUsage/>	addressUsage - Defines a purpose for the address
<effectiveDates>	startDate - Start of effectivity of the address
<startDate/>	endDate - End of effectivity of the address
<endDate/>	seasonalStart - Start of effectivity for a season
<seasonStart/>	seasonalEnd - End of effectivity for a season
<seasonEnd/>	name - Taxpayer name
</effectiveDates>	Contains the address information.
<address>	
<name/>	
<country/>	
<address1/>	
<address2/>	
<address3/>	
<address4/>	
<streetNumber1/>	
<streetNumber2/>	
<county/>	
<city/>	
<state/>	
<postal/>	
<houseType/>	
<latitude/>	
<longitude/>	
<inCityLimit/>	
<comments/>	
</address>	
</location>	

Get Payment History (TSGetPaymentHistory)

These are the node elements of the Get Payment History WSDL.

Node	Description
<TSGetPaymentHistory>	This is the main node of the Get Payment History WSDL. Its dateTimeTagFormat property is set to 'xsd'.
<head>	This node is a sub-node of <TSGetPaymentHistory>. It contains the access keys and audit information of the request. Refer to *Access Keys and Audit Node*.
<lineOfBusiness>	This node is a sub-node of < TSGetPaymentHistory>. It contains the line of business of the request.
<requestStatus> **<errorMessage>**	These nodes are sub-nodes of <TSGetPaymentHistory>. They hold the details of the error message thrown by the web service. Refer to *Error Message Node*.
<customInfo>	This node is a sub-node of <TSGetPaymentHistory>. It holds the holds the details of any customization incorporated into the web service. Refer to *Custom Information Node*.

Node	Description
<requestData>	This node is a sub-node of <TSGetPaymentHistory>. It contains the input data of the request.
<fromDate/> **<toDate/>**	This node is a sub-node of <requestData>. It may contain the date range to filter the payment history retrieval.
<responseData>	This node is a sub-node of <TSGetPaymentHistory>. It contains the response of the request.
<paymentList> **<paymentId/>** **<amount/>** **<date/>** **<currency/>** **<status/>** **</paymentList>**	This list node is a sub-node of < responseData>. It contains payment records of the taxpayer, all or for a specified period. It holds the following information: paymentId - Contains unique identifier of the payment in the revenue management system amount - Contains the payment amount date - Contains the payment date currency - Contains the currency of the amount status - Contains the status of the payment

Get Filing History (TSGetFilingHistory)

These are the node elements of the Get Filing History WSDL.

Node	Description
<TSGetFilingHistory>	This is the main node of the Get Filing History WSDL. Its dateTimeTagFormat property is set to 'xsd'.
<head>	This node is a sub-node of <TSGetFilingHistory>. It contains the access keys and audit information of the request. Refer to *Access Keys and Audit Node*.
<lineOfBusiness>	This node is a sub-node of <TSGetFilingHistory>. It contains the line of business of the request.
<requestStatus> **<errorMessage>**	These nodes are sub-nodes of <TSGetFilingHistory>. They hold the details of the error message thrown by the web service. Refer to *Error Message Node*.
<customInfo>	This node is a sub-node of <TSGetFilingHistory>. It holds the holds the details of any customization incorporated into the web service. Refer to *Custom Information Node*.
<requestData>	This node is a sub-node of <TSGetFilingHistory>. It contains the input data of the request.
<fromDate/> **<toDate/>** **<taxType/>**	This node is a sub-node of <requestData>. It may contain the date range and/or tax type to filter the payment history retrieval.
<responseData>	This node is a sub-node of <TSGetFilingHistory>. It contains the response of the request.
<taxForm> **<formType/>** **<overrideDescription/>** **<formId/>** **<filingPeriodStartDate/>** **<filingPeriodEndDate/>** **<taxType/>** **<filingStatus/>** **<confirmationId/>** **<formSource/>** **<dueDate/>**	This list node is a sub-node of < responseData>. It contains tax filing records of the taxpayer, all or for a specified period. It holds the following information: formType - Contains the type of form overrideDescription - Contains the description for override formId - Contains the unique identifier for forms in the revenue management system filingPeriodStartDate - Contains the start of filing period filingPeriodEndDate - Contains the end of filing period taxType - contains the tax type of the tax role filingStatus - contains the derived filing status for the portal dueDate - contains the due date for filing form

Node	Description
<receivedDate/>	receivedDate - contains the date form was received
<amountDue/>	amountDue - contains the amount due with P&I forecasting of the obligation
<currency/> <documentLocatorNumber/>	currency - Contains the currency code of the amount
<documentLocation/>	documentLocatorNumber - Contains the related document locator number
</taxForm>	documentLocation - Contains the location to the printable tax form

Get Tax Account Alerts (TSGetTaxAccountAlerts)

These are the node elements of the Get Tax Account Alerts WSDL.

Node	Description
<TSGetTaxAccountAlerts>	This is the main node of the Get Tax Account Alerts WSDL. Its dateTimeTagFormat property is set to 'xsd'.
<head>	This node is a sub-node of <TSGetTaxAccountAlerts>. It contains the access keys and audit information of the request. Refer to *Access Keys and Audit Node*.
<lineOfBusiness>	This node is a sub-node of <TSGetTaxAccountAlerts>. It contains the line of business of the request.
<requestStatus> **<errorMessage>**	These nodes are sub-nodes of <TSGetTaxAccountAlerts>. They hold the details of the error message thrown by the web service. Refer to *Error Message Node*.
<customInfo>	This node is a sub-node of <TSGetTaxAccountAlerts>. It holds the holds the details of any customization incorporated into the web service. Refer to *Custom Information Node*.
<requestData>	This node is a sub-node of <TSGetTaxAccountAlerts>. It contains the input data of the request.
<includeAccountAlerts/> **<includeGeneralAlerts/>**	This node is a sub-node of <requestData>. It contains the indicators if account and/or general alerts are going to be retrieved.
<responseData>	This node is a sub-node of <TSGetTaxAccountAlerts>. It contains the response of the request.
<alert> **<alertType/>** **<alertParameters>** **<sequence/>** **<value/>** **</alertParameters>** **<overrideAlertText/>** **<overrideNavigation URL/>** **<documentLocation/>** **</alert>**	This list node is a sub-node of < responseData>. It contains the alerts for the account. It holds the following information: alertType - Contains the type of alert overrideAlertText - Contains the message text to override the portal message overrideNavigation - Contains the url to override navigation documentLocation - Contains the url for the document Holds also the list of alert parameters for the message. Parameter list contains: sequence - Contains the order of the alert parameter value - Contains the value of the alert parameter

Get Taxpayer Summary (TSGetTaxpayerSummary)

These are the node elements of the Get Taxpayer Summary WSDL.

Node	Description
<TSGetTaxpayerSummary>	This is the main node of the Get Taxpayer Summary WSDL. Its dateTimeTagFormat property is set to 'xsd'.
<head>	This node is a sub-node of < TSGetTaxpayerSummary>. It contains the access keys and audit information of the request. Refer to *Access Keys and Audit Node*.
<lineOfBusiness>	This node is a sub-node of < TSGetTaxpayerSummary>. It contains the line of business of the request.
<requestStatus> **<errorMessage>**	These nodes are sub-nodes of < TSGetTaxpayerSummary>. They hold the details of the error message thrown by the web service. Refer to *Error Message Node*.
<customInfo>	This node is a sub-node of < TSGetTaxpayerSummary>. It holds the holds the details of any customization incorporated into the web service. Refer to *Custom Information Node*.
<responseData>	This node is a sub-node of < TSGetTaxpayerSummary>. It contains the response of the request.
<name/> **<taxpayerType/>**	These nodes are sub-nodes of < responseData>. It provides details on the taxpayer such as name and type.
<summaryTitleParameters> **<sequence >** **<value/>** **</summaryTitleParameters>** **<summaryTextParameters>** **<sequence >** **<value/>** **</summaryTextParameters>** **<overrideSummaryTitle/>** **<overrideSummaryText/>**	This node is a sub-node of <responseData>. It holds the parameters for the title and detail messages configured in the portal. sequence - Contains the order of the parameter value - Contains the value of the parameter It may also contain the override messages to replace the portal configuration. overrideSummaryTitle - Contains override message for title override SummaryText - Contains override message for text
<primaryContact> **<contactType/>** **<contactName/>** **<emailAddress/>** **</primaryContact>**	This node is a sub-node of <responseData>. It contains information on the primary contact. contactType - Contains the relationship to main taxpayer contactName - Contains name of contact retrieved based on configured name type emailAddress - Contains email address of the contact

Get Taxpayer Contact Information (TSGetTaxpayerContactInformation)

These are the node elements of the Get Taxpayer Contact Information WSDL.

Node	Description
<TSGetTaxpayerContactInformation>	This is the main node of the Get Taxpayer Contact Information WSDL. Its dateTimeTagFormat property is set to 'xsd'.
<head>	This node is a sub-node of <TSGetTaxpayerContactInformation >. It contains the access keys and audit information of the request. Refer to *Access Keys and Audit Node*.
<lineOfBusiness>	This node is a sub-node of <TSGetTaxpayerContactInformation >. It contains the line of business of the request.
<requestStatus> **<errorMessage>**	These nodes are sub-nodes of <TSGetTaxpayerContactInformation >. They hold the details of the error message thrown by the web service. Refer to *Error Message Node*.

Node	Description
<confirmationData>	This node is a sub-node of <TSGetTaxpayerContactInformation >. It holds the details of the confirmation message returned by the web service. Refer to *Confirmation Message Node*.
<customInfo>	This node is a sub-node of <TSGetTaxpayerContactInformation>. It holds the holds the details of any customization incorporated into the web service. Refer to *Custom Information Node*.
<responseData>	This node is a sub-node of <TSGetTaxpayerContactInformation>. It contains the request and response of the request.
<emailId/>	This node is a sub-node of <responseData>. It contains the new email address in the case of update, or the retrieved email address in the case of read.
<phones> **<sequence/>** **<phoneType/>** **<phoneNumber/>** **<extension/>** **</phones>**	This node is a sub-node of <responseData>. It holds the updated list of phone numbers in the case of update, or the retrieved phones in the case of read. sequence - Contains the order of the parameter phoneType - Contains the type of contact number phoneNumber - Contains the contact number extension - Contains the extension number

Get Taxpayer Correspondence Information (TSGetTaxpayerCorrespondenceInformation)

These are the node elements of the Get Taxpayer Contact Information WSDL.

Node	Description
<TSGetTaxpayerCorrespondenceInformation>	This is the main node of the Get Taxpayer Contact Information WSDL. Its dateTimeTagFormat property is set to 'xsd'.
<head>	This node is a sub-node of <TSGetTaxpayerCorrespondenceInformation>. It contains the access keys and audit information of the request. Refer to *Access Keys and Audit Node*.
<lineOfBusiness>	This node is a sub-node of <TSGetTaxpayerCorrespondenceInformation>. It contains the line of business of the request.
<requestStatus> **<errorMessage>**	These nodes are sub-nodes of <TSGetTaxpayerCorrespondenceInformation>. They hold the details of the error message thrown by the web service. Refer to *Error Message Node*.
<confirmationData>	This node is a sub-node of <TSGetTaxpayerCorrespondenceInformation>. It holds the details of the confirmation message returned by the web service. Refer to *Confirmation Message Node*.
<customInfo>	This node is a sub-node of <TSGetTaxpayerCorrespondenceInformation>. It holds the holds the details of any customization incorporated into the web service. Refer to *Custom Information Node*.
<responseData>	This node is a sub-node of <TSGetTaxpayerCorrespondenceInformation >. It contains the response of the request.
<location> **<sequence/>** **<allowEdit/>** **<allowDelete/>** **<addressId/>** **<addressUsage/>**	This node is a sub-node of <responseData>. It holds the list of addresses of the taxpayer. sequence - Contains the ordering of addresses allowEdit - Indicates whether address is editable allowDelete - Indicates whether address may be deleted

Node	Description
<effectiveDates>	addressId - Uniquely identifies the address in the revenue management system
<startDate/>	addressUsage - Defines a purpose for the address
<endDate/>	startDate - Start of effectivity of the address
<seasonStart/>	endDate - End of effectivity of the address
<seasonEnd/>	seasonalStart - Start of effectivity for a season
</effectiveDates>	seasonalEnd - End of effectivity for a season
<address>	name - Taxpayer name
<name/>	country - Contains the country
<country/>	address1 - Contains the address
<address1/>	address2 - Contains the address
<address2/>	address3 - Contains the address
<address3/>	address4 - Contains the address
<address4/>	streetNumber1 - Contains the street number
<streetNumber1/>	streetNumber2 - Contains the street number
<streetNumber2/>	county - Contains the county
<county/>	city - Contains the city
<city/>	state - Contains the state
<state/>	postal - Contains the postal code
<postal/>	houseType - Contains the type of houde
<houseType/>	latitude - Contains geographic coordinate north-to-south position of the address
<latitude/>	longitude - Contains the geographic coordinate east-to-west position of the address
<longitude/>	inCityLimit - Contains the indicator if within city
<inCityLimit/>	comments - Contains comments
<comments/>	overrideFormattedAddress - Contains the address to override in portal
</address>	
<overrideFormattedAddress/>	
</location>	

Taxpayer Address Maintenance (TSAddressMaintenance)

These are the node elements of the Taxpayer Address Maintenance WSDL.

Node	Description
<TSAddressMaintenance>	This is the main node of the Taxpayer Address Maintenance WSDL. Its dateTimeTagFormat property is set to 'xsd'.
<head>	This node is a sub-node of <TSAddressMaintenance>. It contains the access keys and audit information of the request. Refer to *Access Keys and Audit Node*.
<lineOfBusiness>	This node is a sub-node of <TSAddressMaintenance>. It contains the line of business of the request.
<requestStatus> **<errorMessage>**	These nodes are sub-nodes of <TSAddressMaintenance>. They hold the details of the error message thrown by the web service. Refer to *Error Message Node*.
<confirmationData>	This node is a sub-node of <TSAddressMaintenance>. It holds the details of the confirmation message returned by the web service. Refer to *Confirmation Message Node*.
<customInfo>	This node is a sub-node of <TSAddressMaintenance>. It holds the holds the details of any customization incorporated into the web service. Refer to *Custom Information Node*.

Node	Description
<requestData>	This node is a sub-node of <TSAddressMaintenance>. It contains the input data of the request.
<addressChangeReason/> **<location>** **<addressId/>** **<addressUsage/>** **<effectiveDates>** **<startDate/>** **<endDate/>** **<seasonStart/>** **<seasonEnd/>** **</effectiveDates>** **<address>** **<name/>** **<country/>** **<address1/>** **<address2/>** **<address3/>** **<address4/>** **<streetNumber1/>** **<streetNumber2/>** **<county/>** **<city/>** **<state/>** **<postal/>** **<houseType/>** **<latitude/>** **<longitude/>** **<inCityLimit/>** **<comments/>** **</address>** **</location>**	This node is a sub-node of <requestData>. It contains the address to be added or updated for the taxpayer. addressId - Contains the unique identifier of the address in the revenue management system addressUsage - Contains the purpose for the address startDate - Contains the start of effectivity of the address endDate - Contains the end of effectivity of the address seasonalStart - Contains the start of effectivity for a season seasonalEnd - Contains the end of effectivity for a season name - Containst the taxpayer name country - Contains the country address1 - Contains the address address2 - Contains the address address3 - Contains the address address4 - Contains the address streetNumber1 - Contains the street number streetNumber2 - Contains the street number county - Contains the county city - Contains the city state - Contains the state postal - Contains the postal code houseType - Contains the type of houde latitude - Contains geographic coordinate north-to-south position of the address longitude - Contains the geographic coordinate east-to-west position of the address inCityLimit - Contains the indicator if within city comments - Contains comments

Enrollment Service Request (TSEnrollmentServiceRequest) - WSDL

These are the node elements of the Prepare External Payment Data WSDL.

Node	Description
<TSEnrollmentServiceRequest>	This is the main node of the Generic Service Request WSDL. Its dateTimeTagFormat property is set to 'xsd'.
<head>	This node is a sub-node of <TSEnrollmentServiceRequest>. 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 <TSEnrollmentServiceRequest>. They hold the details of the error message thrown by the web service. Refer to *Error Message Node*.

Node	Description
<linkedRequest>	This node is a sub-node of <TSEnrollmentServiceRequest>. It may contain the reference number identifying Taxpayer Identification performed for this payment
<confirmationData>	This node is a sub-node of <TSEnrollmentServiceRequest>. It holds the details of the confirmation message returned by the web service. Refer to *Confirmation Message Node*.
<mainData>	This node is a sub-node of <TSEnrollmentServiceRequest>. It contains the main data in the request.
<serviceRequestType>	This node is a sub-node of <mainData>. It contains the type of Service Request that was submitted.
<responseMode>	This node is a sub-node of <mainData>. It holds the mode of response that is associated with the request. Value can either be Synchronous or Asynchronous.
<serviceRequestData> **<enrollmentID>** **<lineOfBusiness>** **<requestField>** **<sequence>** **<fieldName>** **<fieldValue>** **</requestField>** **</serviceRequestData>**	This node is a sub-node of <mainData>. It holds: **Line of Business** and **Enrollment ID**. This pair of values should be captured on the enrollment event record and uniquely identify enrollment event in the back-end system. A list of service request fields. <requestField> - Each sub-node holds the information of a specific field in the request. <sequence> - Contains the field's sequence number. <fieldName> - Contains the name of a field. <fieldValue> - Contains the value of a field.
<customInfo>	This node is a sub-node of <TSEnrollmentServiceRequest>. It holds the holds the details of any customization incorporated into the web service. Refer to *Custom Information Node*.
<userAccess type="group"> **<accessEntity type="list">** **<sequence/>** **<lineOfBusiness/>** **<revenueMgmtSystem/>** **<enrollmentId/>** **<status/>** **<accessType/>** **<ACCESS KEYS 1 - 10>** **</accessEntity>** **</userAccess>**	This node is a sub-node of <TSEnrollmentServiceRequest>. It holds one or more access keys that represent the taxpayer's information in the revenue management system.

Get Enrollment Summary (TSGetEnrollmentSummary) - WSDL

These are the node elements of the Get Enrollment Summary message.

Node	Description
<TSGetEnrollmentSummary>	This is the main node of the request. Its dateTimeTagFormat property is set to 'xsd'.
<head>	This node is a sub-node of <TSGetEnrollmentSummary>. 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 <TSGetRefundStatus>. They hold the details of the error message thrown by the web service. Refer to *Error Message Node*.

Node	Description
<linkedRequest>	This node is a sub-node of < TSGetEnrollmentSummary>. Part of the common message fragment, not in use.
<confirmationData>	This node is a sub-node of <TSGetEnrollmentSummary>. It may hold the details of the confirmation message returned by the web service. Refer to *Confirmation Message Node*. Not in use.
<mainData>	This node is a sub-node of <TSGetEnrollmentSummary>. It contains the main data of the request.
<userAccess>	This node is a sub-node of <mainData>. It contains the list of access "units".
<accessEntity>	This is a list. Each element contains the data related to the single access keys set.
<sequence/> **<taxpayerName/>** **<accountName/>** **<isDefaultAccessEntry/>** **<lineOfBusiness/>** **<revenueMgmtSystem/>** **<enrollmentId/>** **<status/>** **<accessType/>** **<key1>** **<name/>** **<value/>** **</key1>** **<key2>** **<name/>** **<value/>** **</key2> ....** **.....** **<key10>** **<name/>** **<value/>** **</key10>** **<summaryTitle>** **<parameters>** **<sequence/>** **<value/>** **</parameters>** **</summaryTitle>** **<summaryDetail>** **<parameters>** **<sequence/>** **<value/>** **</parameters>** **</summaryDetail>**	The elements in bold are populated in the integration layer with data queried from the User Access store. The rest is populated by the revenue management system and returned with the response.
<customInfo>	This node is a sub-node of <TSGetEnrollmentSummary>. It holds the details of any customization incorporated into the web service. Refer to *Custom Information Node*.

User Enrollment (TSGetUserEnrollment) - WSDL

These are the node elements of the TSGetUserEnrollment message.

Node	Description
<TSGetUserEnrollment>	This is the main node of the Get User Enrollment Request WSDL. Its dateTimeTagFormat property is set to 'xsd'.
<head>	This node is a sub-node of <TSGetUserEnrollment>. 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 < TSGetUserEnrollment >. They hold the details of the error message thrown by the web service. Refer to *Error Message Node*.
<linkedRequest>	This node is a sub-node of <TSGetUserEnrollment>. Part of the common message fragment, and not in use.
<confirmationData>	This node is a sub-node of <TSGetUserEnrollment>. It holds the details of the confirmation message returned by the web service. Refer to *Confirmation Message Node*.
<mainData>	This node is a sub-node of <TSGetUserEnrollment>. It contains the main data of the request.
<isUserEnrolled>	This node is a sub-node of <mainData>. Indicates whether user already has at least one Approved record in User Access Store.
<newKeysExists>	This node is a sub-node of <mainData>. Indicates whether revenue management found new information and granted the access to it (for implicit enrollment).
<enrollmrentSummary> **<enrollmentEvent>** **<lineOfBusiness>** **<enrollmentId>** **</enrollmentEvent>** **</ enrollmrentSummary>**	This node is a sub-node of <mainData>. It provides the snapshot of user enrollment(what Lines of Business have already been enrolled, along with the corresponding enrollment ID).
<userAccess>	This node is a sub-node of <mainData>. It contains the list of access "units"
<accessEntity>	This is a list. Each element contains the data related to the single access keys set.
<sequence/> **<lineOfBusiness/>** **<revenueMgmtSystem/>** **<enrollmentId/>** **<status/>** **<accessType/>** **<key1>** **<name/>** **<value/>** **</key1>** **<key2>** **<name/>** **<value/>** **</key2> ....** **.....** **<key10>** **<name/>**	Access entity information includes access type, access keys, and line of business. On the request it is populated with existing records queried from the User Access store. On the response, it is populated with new access keys determined in the revenue management system (implicit enrollment refresh).

Node	Description
<value/>	
</key10>	
<customInfo>	This node is a sub-node of <TSGetUserEnrollment>. It holds the holds the details of any customization incorporated into the web service. Refer to *Custom Information Node*.

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.
<linkedRequest>	This node is a sub node of **<TSTaxpayerServiceRequest>**. It may contain the reference number identifying Taxpayer Identification performed for this form processing.
<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>** **<fieldValue>** **</requestField>** **</serviceRequestData>**	This node is a sub node of **<mainData>**. It holds a list of service request fields. **requestField** – a list of service reques fields **sequence** contains the field's sequence number. **fieldName** contains the name of a field. **fieldValue** contains the value of a field.
<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 detals.

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> **<errorMessage>**	These nodes are sub nodes of **<TSTaxpayerIdentification>**. 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 **<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> **<requestField>** **<sequence>** **<fieldName>** **<fieldValue>** **</requestField>** **</serviceRequestData>**	This node is a sub node of <mainData>. It holds a list of service request fields. **requestField** – a list of service reques fields **sequence** contains the field's sequence number. **fieldName** contains the name of a field. **fieldValue** contains the value of a field.
<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 detals.
<responseData> **<responseDetails>** **<sequence>** **<taxpayerId>** **</responseDetails>** **</responseData>**	This node is a sub node of **<TSTaxpayerIdentification>**. It holds response details from the web service. **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.

Refund Inquiry Request - TSGetRefundStatus

Node	Description
<TSGetRefundStatus>	This is the main node of the Refund Inquiry Service Request WSDL. **dateTimeTagFormat** property value is set to **'xsd'.**
<head>	This node is a sub node of **<TSGetRefundStatus>**. 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 **<TSGetRefundStatus>**. Refer to Error Message Node for details.
<confirmationData>	This node is a sub node of <TSGetRefundStatus>. Refer to Confirmation Message Node for details.
<linkedRequest>	This node is a sub node of <TSGetRefundStatus>. It may contain the reference number identifying Taxpayer Identification performed for this form processing.
<mainData>	This node is a sub node of **<TSGetRefundStatus>**. It contains the main data of the request.
<serviceRequestType>	This node is a sub node of <mainData>. Service Request Type code, alphanumeric.
<serviceRequestData> **<requestField>** **<sequence>** **<fieldName>** **<fieldValue>** **</requestField>** **</serviceRequestData>**	This node is a sub node of <mainData>. 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.
<responseData> **<responseDetails>** **<expectedDate>** **<refundAmount>** **<carryOverAmount>** **<offsetAmount>** **</responseDetails>** **</responseData>**	This node is a sub node of **<TSGetRefundStatus>**. 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 are any). **offsetAmount** contains the offset amount (if there's any).
<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 detals.

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 <TSOneTimePayment>. Refer to Confirmation Message Node for details.
<linkedRequest>	This node is a sub node of <TSOneTimePayment>. It may contain the reference number identifying Taxpayer Identification performed for this form processing.
<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.

Node	Description
<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 the Custom Information Node for details.

Prepare Payment Data - TSPrepareExtPaymentData

Node	Description
<TSPrepareExtPaymentData>	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 **<TSPrepareExtPaymentData>**. It contains the access keys and the audit information of the request. The valid vaues for the **<action>** element are: VALIDATEONLY and PREPARE. Refer to Access Keys and Audit Node for moe details.
<requestStatus> **<errorMessage>**	These nodes are sub nodes of **<TSPrepareExtPaymentData>**. Refer to Error Message Node for details
<confirmationData>	This node is a sub node of <**TSPrepareExtPaymentData**>. Common fragment, not in use for this web service
<mainData>	This node is a sub node of **<TSPrepareExtPaymentData>**. 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>**	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.

Node	Description
</destinationDetails>	**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.
<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 detals.

Process Reconciliation Report - TSProcessExtPayReportRecord

Node	Description
<TSProcessExtPayReportRecord>	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 **<TSProcessExtPayReportRecord>**. It contains the access keys and the audit information of the request. Refer to Access Keys and Audit Node for moe details..
<requestStatus> **<errorMessage>**	These nodes are sub nodes of **<TSProcessExtPayReportRecord>**. Refer to Error Message Node for details
<confirmationData>	This node is a sub node of **<TSProcessExtPayReportRecord>**. Common fragment, not in use by this web service
<paymentReportRecord>	This node is a sub node of **<TSProcessExtPayReportRecord>**. 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

Node	Description
<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 pament information.
<customInfo>	This node is a sub node of <TSProcessExtPayReportRecord>. It holds the 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.

Address Data Common

This is the common node that contains address details.

Node	Description
<addressId>	**addressed** contains the unique identifier of this address in the back-end system
<addressUsage>	This node is a sub node of **<head>**. It indicates the specific business purpose of the request. Optional.
<effectiveDates> **<startDate/>** **<endDate/>** **<seasonalStart/>** **<seasonalEnd/>** **</effectiveDates>**	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)
<address>	This node is contains the address details.
<name/>	**name** contains the entity name of the taxpayer
<country/>	**country** contains the country code
<address1/>	**address1** contains specific details such as number, lot, building, street
<address2/>	**address2** additional node for specific details such as number, lot, building, street
<address3/>	**address3** additional node for specific details such as number, lot, building, street
<address4/>	**address4** additional node for specific details such as number, lot, building, street
<streetNumber1/>	**streetNumber1** contains the street and number
<streetNumber2/>	**streetNumber2** additional node for street and number
<county/>	**county** contains the county value
<city/>	**city** contains the city
<state/>	**state** contains the state code
<postal/>	**postal** contains the postal code
<houseType/>	**houseType** contains the house type (House Boat Reference, Trailer Reference)
<latitude/>	**latitude** contains the geographic coordinate that specifies the north-south position
<longitude/>	**longitude** contains the geographic coordinate that specifies the east-west position
<inCityLimit/>	**inCityLimit** contains the indicator if the address is within city limit
<comments/>	**comments** contains additional information on the address
</address>	

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.
<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.
<key6> **<name>** **<value>** **</key6>**	This node is a sub node of **<head>**. Contains the access key for the service: account ID, taxpayer ID, tax type, etc.
<key7> **<name>** **<value>** **</key7>**	This node is a sub node of **<head>**. Contains the access key for the service: account ID, taxpayer ID, tax type, etc.
<key8> **<name>** **<value>** **</key8>**	This node is a sub node of **<head>**. Contains the access key for the service: account ID, taxpayer ID, tax type, etc.
<key9> **<name>** **<value>** **</key9>**	This node is a sub node of **<head>**. Contains the access key for the service: account ID, taxpayer ID, tax type, etc.

Node	Description
<key10> **<name>** **<value>** **</key10>**	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.
<accessType>	This node is a sub node of **<head>**. It contains the access type that identifies a high level entity in the revenue management system to which the user has access.

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> **<messageCategory>** **<messageNumber>** **<messageParameters>** **<parameters>** **<sequence>** **<parameterType>** **<parameterValue>** **</parameters>** **</messageParameters>** **<currency>** **<messageTxtOvrd>** **</header>**	This node is a sub node of <confirmationData>. It contains a list of the header information of the confirmation message. **messageCategory** contains the Message Category number of the confirmation. **messageNumber** contains the Message Number **messageParameters** contains Message Parameters **parameters** parameters list **sequence** contains the sequence number of the parameter. **parameterType** contains the type of the parameter. Valid values: DATE (Date), STRING (String), NUMBER (Number) or CURRENCY (Currency). **parameterValue** contains the value of the parameter. **currency** contains the currency code to be used in the message when translating parameters of type CURRENCY **messageTxtOvrd** contains the override message text.
<details> **<messageCategory>** **<messageNumber>** **<messageParameters>** **<parameters>** **<sequence>** **<parameterType>** **<parameterValue>** **</parameters>** **<currency>** **<messageTxtOvrd>** **</details>**	This node is a sub node of **<confirmationData>**. It contains a list of confirmation details (messages). **messageCategory** contains the Message Category number of the confirmation. **messageNumber** contains the Message Number **messageParameters** contains Message Parameters **parameters** parameters list **sequence** contains the sequence number of the parameter. **parameterType** contains the type of the parameter. Valid values: DATE (Date), STRING (String), NUMBER (Number) or CURRENCY (Currency). **parameterValue** contains the value of the parameter. **currency** contains the currency code to be used in the message when translating parameters of type CURRENCY **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> **<sequence>** **<fieldName>**	This node contains a list of generic field name/value pairs. It is a placeholder for the data populated by various customization extensions. **sequence** contains the sequence number of the entry.

Node	Description
<fieldValue>	**fieldName** contains the name of the field.
</customInfo>	**fieldValue** contains the value

Print Document (TSPrintDocument)

The following table describes the node elements of the Print Document message.

Node	Description
<TSPrintDocument>	This is the main node of the Print Document WSDL. **dateTimeTagFormat** property value is set to **'xsd'.**
<head>	This node is a sub node of **<TSPrintDocument>**. 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 **<TSPrintDocument>**. Refer to Error Message Node for details.
<confirmationData>	This node is a sub node of **<TSPrintDocument>.** Refer to Confirmation Message Node for details.
<linkedRequest>	This node is a sub-node of **<TSPrintDocument>**. Part of the common message fragment, and not in use.
<input>	This node is a sub node of **<TSPrintDocument>**. It contains the input data of the request.
<documentSource>	This node is a sub node of **<input>**. It contains the Form Category with values could be either TAXFORM or REGFORM.
<documentData>	This node is a sub node of **< input >**. It contains the actual form data which to be formatted to generate a printable document.
<output>	This node is a sub node of **<TSPrintDocument>**. It contains the output data of the request.
<documentContent>	This node is a sub node of **<output>. It** contains the base - 64 encoded data of the input document data.
<documentType>	This node is a sub node of **<output>** and contains the file type the input document data was transformed to.
<customInfo>	This node is a sub node of **< TSPrintDocument >**. 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.

Import Form Definition (TSRetrieveFormTypeDefinitions) - WSDL

The following table describes the node elements of the Import Form Definition message.

Node	Description
<TSRetrieveFormTypeDefinitions>	This is the main node of the Import Form Definition WSDL. **dateTimeTagFormat** property value is set to **'xsd'.**
<head>	This node is a sub node of **<TSRetrieveFormTypeDefinitions>**. 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 **<TSRetrieveFormTypeDefinitions>**.
<errorMessage>	Refer to Error Message Node for details.
<confirmationData>	This node is a sub node of **<TSRetrieveFormTypeDefinitions>.** Refer to Confirmation Message Node for details.

Node	Description
<mainData>	This node is a sub node of **<TSRetrieveFormTypeDefinitions>**. It contains the main data of the request.
<input>	This node is a sub node of **<main>**. It contains the input data of the request.
<formType>	This node is a sub node of **<input> and** contains the form type code
<language>	This node is a sub node of **<input> and** contains the specific language that that the form represented by form type will be imported
<output>	This node is a sub node of **<main>**. It contains the output data of the request.
<formDef/> **<formLookups/>** **<availableLanguages/>**	These nodes are sub-nodes of **<output>** that holds the structure of the imported form. It contains the following information: formDef - Contains the structure of the import form formLookups - Contains the lookups available on the form availableLanguages - Contains the list of languages to which the form can be imported to.
<customInfo>	This node is a sub node of **< TSRetrieveFormTypeDefinitions >**. 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.

Refresh Lookup (TSRefreshFormLookup) - WSDL

These are the node elements of the Refresh Lookup message.

Node	Description
<TSRefreshFormLookup>	This is the main node of the Refresh Form Lookup WSDL. **dateTimeTagFormat** property value is set to **'xsd'.**
<head>	This node is a sub node of **<TSRefreshFormLookup>**. 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 **<TSRefreshFormLookup>**. Refer to Error Message Node for details.
<confirmationData>	This node is a sub node of **<TSRefreshFormLookup>.** Refer to Confirmation Message Node for details.
<mainData>	This node is a sub node of **<TSRefreshFormLookup>**. It contains the main data of the request.
<input>	This node is a sub node of **<mainData>**. It contains the input data of the request.
<lookupName>	This node is a sub node of **<input>** and contains the lookup name where values will be collected
<output>	This node is a sub node of **<TSRefreshFormLookup>**. It contains the output data of the request.
<values>	This node is a sub node of **<output>**. It contains the lookup values available for the specific lookup.
<value> **<description>**	These nodes are sub-nodes of **<values>** that holds the following information: value - Contains the lookup value code description - Contains the lookup value descriptions in various languages
<customInfo>	This node is a sub node of **< TSRefreshFormLookup >**. 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.

Retrieve Active Form Types (TSRetrieveActiveFormTypes)

The following table describes the node elements of the Retrieve Active Form Types message.

Node	Description
<TSRetrieveActiveFormTypes>	This is the main node of the Retrieve Active Form Types WSDL. **dateTimeTagFormat** property value is set to **'xsd'.**
<head>	This node is a sub node of **<TSRetrieveActiveFormTypes>**. 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 **<TSRetrieveActiveFormTypes>**. Refer to Error Message Node for details.
<confirmationData>	This node is a sub node of **<TSRetrieveActiveFormTypes>.** Refer to Confirmation Message Node for details.
<mainData>	This node is a sub node of **<TSRetrieveActiveFormTypes>**. It contains the main data of the request.
<formSubType> **<language>** **<startDate>** **<endDate>** **<description>**	These nodes are sub-nodes of **<mainData>** that holds the following information: formSubType - Contains the form category of the form types that will be retrieved. Valid values are TAXFORM and REGFORM. language - Contains the language of the description for the form types returned startDate - Filters the form types to be active since this date endDate - Filters the form types to be active before this date description - It filters the form types to match this description. The symbol % can be used as wild character.
<forms>	This node is a sub node of **<mainData>.** It contains the retrieved active form types.
<formType> **<startDate>** **<endDate>** **<description>**	These nodes are sub-nodes of **<mainData>** that holds the following information: formType - Contains the form type code startDate - Contains the form type start date endDate - Contains the form type start date description - Contains the description within the specified language in the input
<customInfo>	This node is a sub node of **< TSRetrieveActiveFormTypes >**. 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.

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>
```

Document Locator Number Service

DocumentLocatorNumberService Action CREATE Request

```
<process xmlns="http://xmlns.oracle.com/OTSS/OTSSDocumentLocatorNumberService/
OTSSDocumentLocatorNumberBPELProcess">
<action name="DLN.Action.Create">CREATE</action>
<prefix name="TaxForm.DLN.Prefix">TFDLN</prefix>
<documentLocatorNumber/>
</process>
```

DocumentLocatorNumberService Action CREATE Response

```
<processResponse xmlns:client="http://xmlns.oracle.com/OTSS/
OTSSDocumentLocatorNumberService/OTSSDocumentLocatorNumberBPELProcess"
 xmlns="http://xmlns.oracle.com/OTSS/OTSSDocumentLocatorNumberService/
OTSSDocumentLocatorNumberBPELProcess"><client:documentLocatorNumber>TFDLN00000005090</
client:documentLocatorNumber>
<client:valid/>
</processResponse>
```

DocumentLocatorNumberService Action VALIDATE Request

```
<process xmlns="http://xmlns.oracle.com/OTSS/OTSSDocumentLocatorNumberService/
OTSSDocumentLocatorNumberBPELProcess">
<action name="DLN.Action.Validate">VALIDATE</action>
<prefix/>
<documentLocatorNumber>TFDLN00000004218</documentLocatorNumber>
</process>
```

DocumentLocatorNumberService Action CREATE Response

```
<processResponse xmlns="http://xmlns.oracle.com/OTSS/OTSSDocumentLocatorNumberService/
OTSSDocumentLocatorNumberBPELProcess">
<valid>true</valid>
<documentLocatorNumber/>
</processResponse>
```

GetRefundStatus

GetRefundStatus Request

The following samples illustrate a scenario in which a casual user makes an inquiry about a tax refund status.

```
<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:key6>
<ns0:name/>
<ns0:value/>
</ns0:key6>
<ns0:key7>
<ns0:name/>
<ns0:value/>
</ns0:key7>
<ns0:key8>
<ns0:name/>
<ns0:value/>
</ns0:key8>
<ns0:key9>
<ns0:name/>
<ns0:value/>
</ns0:key9>
<ns0:key10>
<ns0:name/>
<ns0:value/>
</ns0:key10>
<ns0:accessType/>
<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:requestField>
<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:key6>
<ns0:name/>
<ns0:value/>
</ns0:key6>
<ns0:key7>
<ns0:name/>
<ns0:value/>
</ns0:key7>
<ns0:key8>
<ns0:name/>
<ns0:value/>
</ns0:key8>
<ns0:key9>
<ns0:name/>
<ns0:value/>
</ns0:key9>
<ns0:key10>
<ns0:name/>
<ns0:value/>
</ns0:key10>
<ns0:accessType/>
<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:TSOneTimePayment xmlns:ns1="http://oracle.com/TSOneTimePayment.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: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>PSRMSS</ns1:paymentVendor>
</ns1:mainData>
</ns1:TSOneTimePayment>
```

OneTimePayment Response

```
<TSOneTimePayment xmlns:ns0="http://oracle.com/TSOneTimePayment.xsd"
dateTimeTagFormat=""
xmlns="http://oracle.com/TSOneTimePayment.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:key6>
<ns0:name/>
<ns0:value/>
</ns0:key6>
<ns0:key7>
<ns0:name/>
<ns0:value/>
</ns0:key7>
<ns0:key8>
<ns0:name/>
<ns0:value/>
</ns0:key8>
<ns0:key9>
<ns0:name/>
<ns0:value/>
</ns0:key9>
<ns0:key10>
<ns0:name/>
<ns0:value/>
</ns0:key10>
<ns0:accessType/>
<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: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:messageTxtOvrd/>
</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>
</TSOneTimePayment>
```

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: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>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:key6>
<client:key7/>
<client:key8/>
<client:key9/>
<client:key10/>
<client:accessType/>
<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:messageTxtOvrd/>
</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-8497-511137b02a1d</client:externalID>
```

```
<client:feeRequirement/>
</client:mainData>
</TSPrepareExtPaymentData>
```

RetrievePaymentsDue

RetrievePaymentsDue Request

This example illustrates the request for payment destination **OBLIGATION** (pay tax obligation). This request is submitted when an enrolled user signs in, chooses a specific tax account, and then selects the option, ***Pay Tax Obligation*** .

```
<ns1:TSRetrievePaymentsDue xmlns:ns1="http://oracle.com/TSRetrievePaymentsDue.xsd">
<ns1:head>
<ns1:key1>
<ns1:name>PER_ID</ns1:name>
<ns1:value>6638893506</ns1:value>
</ns1:key1>
<ns1:key2>
<ns1:name>TAX_ROLE_ID</ns1:name>
<ns1:value>9037315275</ns1:value>
</ns1:key2>
<ns1:webUserId>USER01</ns1:webUserId>
<ns1:webUserName>John Doe</ns1:webUserName>
<ns1:emailAddress/>
<ns1:ipAddress>10.186.253.49</ns1:ipAddress>
<ns1:accessType>TAXROLE</ns1:accessType>
</ns1:head>
<ns1:mainData>
<ns1:paymentDestination>OBLIGATION</ns1:paymentDestination>
</ns1:mainData>
</ns1:TSRetrievePaymentsDue>
```

RetrievePaymentsDue Response

The response contains details of two payments due for tax obligations. This information is derived using input tax account id. Each entry includes payment amount, payment destination field with tax obligation id and payment summary parameters: tax obligation type code and payment due date.

```
<ns0:TSRetrievePaymentsDue xmlns:ns0="http://oracle.com/TSRetrievePaymentsDue.xsd"
 dateTimeTagFormat="" xmlns="http://oracle.com/TSRetrievePaymentsDue.xsd">
<ns0:head>
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>1237870586</ns0:value>
</ns0:key1>
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>0453614936</ns0:value>
</ns0:key2>
<ns0:key3/>
<ns0:key4/>
<ns0:key5/>
<ns0:key6/>
<ns0:key7/>
<ns0:key8/>
<ns0:key9/>
<ns0:key10/>
<ns0:accessType>TAXROLE</ns0:accessType>
<ns0:webUserId>QAUSER1</ns0:webUserId>
<ns0:webUserName>QAUSER1</ns0:webUserName>
```

```
<ns0:emailAddress/>
<ns0:ipAddress>10.186.252.171</ns0:ipAddress>
</ns0:head>
<ns0:errorMessage>
<ns0:currency/>
<ns0:messageTxtOvrd/>
</ns0:errorMessage>
<ns0:confirmationData>
<ns0:header>
<ns0:messageCategory/>
<ns0:messageTxtOvrd/>
</ns0:header>
</ns0:confirmationData>
<ns0:mainData>
<ns0:paymentDestination>OBLIGATION</ns0:paymentDestination>
<ns0:paymentsDue>
<ns0:amount>10229.72</ns0:amount>
<ns0:destinationFields>
<ns0:sequence>1</ns0:sequence>
<ns0:fieldValue>0453614113</ns0:fieldValue>
</ns0:destinationFields>
<ns0:summary>
<ns0:parameters>
<ns0:sequence>1</ns0:sequence>
<ns0:value>IND-PAYSA</ns0:value>
</ns0:parameters>
<ns0:parameters>
<ns0:sequence>3</ns0:sequence>
<ns0:value>2014-05-31</ns0:value>
</ns0:parameters>
</ns0:summary>
</ns0:paymentsDue>
<ns0:paymentsDue>
<ns0:amount>1689.23</ns0:amount>
<ns0:destinationFields>
<ns0:sequence>1</ns0:sequence>
<ns0:fieldValue>0453623344</ns0:fieldValue>
</ns0:destinationFields>
<ns0:summary>
<ns0:parameters>
<ns0:sequence>2</ns0:sequence>
<ns0:value>IND-PAYSA</ns0:value>
</ns0:parameters>
<ns0:parameters>
<ns0:sequence>3</ns0:sequence>
<ns0:value>2014-04-30</ns0:value>
</ns0:parameters>
</ns0:summary>
</ns0:paymentsDue>
</ns0:mainData>
</ns0:TSRetrievePaymentsDue>
```

RequestStatusInquiry

RequestStatusInquiry Request

The examples below illustrate the **Track Your Transaction** flow. The user provides an input confirmation ID and the system returns the confirmation details associated with this input.

```
<ns1:TSGetConfirmationInformation dateTimeTagFormat="xsd"
xmlns:ns1="http://oracle.com/TSGetConfirmationInformation.xsd">
<ns1:head>
<ns1:key1/>
```

```
<ns1:key2/>
<ns1:key3/>
<ns1:key4/>
<ns1:key5/>
<ns1:key6/>
<ns1:key7/>
<ns1:key8/>
<ns1:key9/>
<ns1:key10/>
<ns1:accessType/>
<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:messageTxtOvrd/>
</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:messageTxtOvrd/>
</ns0:details>
</ns0:confirmationData>
</TSGetConfirmationInformation>
```

TaxClearanceCertificate

TaxClearanceCertificate Request

This example illustrates the generic service request flow. The sample shows the request for a tax clearance certificate submitted by a casual user.

```


<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 Goverment
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:key6/>
<otss:key7/>
<otss:key8/>
<otss:key9/>
<otss:key10/>
<otss:accessType/>
<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:messageTxtOvrd/>
</otss:errorMessage>
<otss:confirmationData>
<otss:confirmationId>SRID81000000</otss:confirmationId>
<otss:header>
<otss:messageCategory>11126</otss:messageCategory>
<otss:messageNumber>10001</otss:messageNumber>
<otss:messageTxtOvrd/>
</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:messageTxtOvrd/>
</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>Requried for Goverment
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: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:messageTxtOvrd />
</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-57306fddbd69</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: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:messageTxtOvrd />
</ns0:errorMessage>
<ns0:paymentReportRecord>
<ns0:paymentVendor>OPC</ns0:paymentVendor>
<ns0:amount>13.00</ns0:amount>
<ns0:paymentDate>2012-08-24</ns0:paymentDate>
<ns0:payDestinationType>C1CN</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>
```

Initial Enrollment Request

The examples below illustrate the initial enrollment request flow.

CreateEnrollment Request (Self Service to Integration Layer)

This example shows an initial enrollment request submitted by user USER01 (John Doe). The user is requesting enrollment for business taxes (Line Of Business BUS).

```
<ns1:TSEnrollmentServiceRequest xmlns:ns1="http://oracle.com/
TSEnrollmentServiceRequest.xsd">
<ns1:head>
<ns1:webUserId>USER01</ns1:webUserId>
<ns1:webUserName>John Doe</ns1:webUserName>
<ns1:ipAddress>10.154.138.130</ns1:ipAddress>
</ns1:head>
<ns1:mainData>
<ns1:serviceRequestType>ENROLL_BUSINESS</ns1:serviceRequestType>
<ns1:responseMode>SYNCH</ns1:responseMode>
<ns1:serviceRequestData>
<ns1:lineOfBusiness>BUS</ns1:lineOfBusiness>
<ns1:requestField>
<ns1:sequence>1</ns1:sequence>
<ns1:fieldName>ID_TYPE_BUS</ns1:fieldName>
```

```
<ns1:fieldValue>SSN</ns1:fieldValue>
</ns1:requestField>
<ns1:requestField>
<ns1:sequence>2</ns1:sequence>
<ns1:fieldName>ID_VALUE</ns1:fieldName>
<ns1:fieldValue>111-00-6644</ns1:fieldValue>
</ns1:requestField>
<ns1:requestField>
<ns1:sequence>3</ns1:sequence>
<ns1:fieldName>BIRTH_DATE</ns1:fieldName>
<ns1:fieldValue>1966-11-12</ns1:fieldValue>
</ns1:requestField>
</ns1:serviceRequestData>
</ns1:mainData>
</ns1:TSEnrollmentServiceRequest>
```

CreateEnrollment Request (Integration Layer to Revenue Management)

Once the request has reached the integration layer, the SOA composite assigns the confirmation ID and enrollment ID.

```
<TSEnrollmentServiceRequest xmlns:params="http://schemas.oracle.com/service/bpel/common"
 xmlns:ns0="http://oracle.com/TSEnrollmentServiceRequest.xsd" dateTimeTagFormat=""
 xmlns="http://oracle.com/TSEnrollmentServiceRequest.xsd">
<ns0:head>
<ns0:key1/>
<ns0:key2/>
<ns0:key3/>
<ns0:key4/>
<ns0:key5/>
<ns0:key6/>
<ns0:key7/>
<ns0:key8/>
<ns0:key9/>
<ns0:key10/>
<ns0:webUserId>USER01</ns0:webUserId>
<ns0:webUserName>John Doe</ns0:webUserName>
<ns0:ipAddress>10.154.138.130</ns0:ipAddress>
</ns0:head>
<ns0:errorMessage>
<ns0:messageParameters/>
</ns0:errorMessage>
<ns0:confirmationData>
<ns0:confirmationId>CEID15410000</ns0:confirmationId>
<ns0:header>
<ns0:messageParameters/>
</ns0:header>
</ns0:confirmationData>
<ns0:mainData>
<ns0:serviceRequestType>ZZ-BUSINESS-ENROLLMENT</ns0:serviceRequestType>
<ns0:responseMode>SYNCH</ns0:responseMode>
<ns0:serviceRequestData>
<ns0:lineOfBusiness>C1BU</ns0:lineOfBusiness>
<ns0:enrollmentId>1aa7720f134c42c98ce9aac9c0a8af3b</ns0:enrollmentId>
<ns0:requestField>
<ns0:sequence>1</ns0:sequence>
<ns0:fieldName>ID_TYPE_BUS</ns0:fieldName>
<ns0:fieldValue>SSN</ns0:fieldValue>
</ns0:requestField>
<ns0:requestField>
<ns0:sequence>2</ns0:sequence>
<ns0:fieldName>ID_VALUE</ns0:fieldName>
<ns0:fieldValue>111-00-6644</ns0:fieldValue>
</ns0:requestField>
<ns0:requestField>
<ns0:sequence>3</ns0:sequence>
<ns0:fieldName>BIRTH_DATE</ns0:fieldName>
<ns0:fieldValue>1966-11-12</ns0:fieldValue>
```

```
</ns0:requestField>
</ns0:serviceRequestData>
</ns0:mainData>
</TSEnrollmentServiceRequest>
```

CreateEnrollment Response (Revenue Management to Integration Layer)

The response from the revenue management system contains tax account identifiers and confirmation information. In this example the response contains information about two tax accounts.

This response is processed in the integration layer as follows:

- The process invokes UserAccesService that verifies number of users that already have access to each tax account.
- New entries added to User Account Access store.
- Tax account identifiers are removed from the message.

```
<TSEnrollmentServiceRequest xmlns="http://oracle.com/TSEnrollmentServiceRequest.xsd"
dateTimeTagFormat="">
<head>
<key1/>
<key2/>
<key3/>
<key4/>
<key5/>
<key6/>
<key7/>
<key8/>
<key9/>
<key10/>
<webUserId>GUEST</webUserId>
<webUserName>GUEST</webUserName>
<ipAddress>10.154.138.130</ipAddress>
</head>
<errorMessage>
<messageParameters/>
</errorMessage>
<confirmationData>
<confirmationId>CEID15410000</confirmationId>
<header>
<messageCategory>11126</messageCategory>
<messageNumber>12012</messageNumber>
</header>
<details>
<sequence>1</sequence>
<messageCategory>11126</messageCategory>
<messageNumber>12013</messageNumber>
</details>
</confirmationData>
<mainData>
<serviceRequestType>ZZ-BUSINESS-ENROLLMENT</serviceRequestType>
<responseMode>SYNCH</responseMode>
<serviceRequestData>
<lineOfBusiness>C1BU</lineOfBusiness>
<enrollmentId>1aa7720f134c42c98ce9aac9c0a8af3b</enrollmentId>
<requestField>
<sequence>1</sequence>
<fieldName>ID_TYPE_BUS</fieldName>
<fieldValue>SSN</fieldValue>
</requestField>
<requestField>
<sequence>2</sequence>
<fieldName>ID_VALUE</fieldName>
<fieldValue>111-00-6644</fieldValue>
</requestField>
<requestField>
```

```
<sequence>3</sequence>
<fieldName>BIRTH_DATE</fieldName>
<fieldValue>1966-11-12</fieldValue>
</requestField>
</serviceRequestData>
</mainData>
<userAccess>
<accessEntity>
<sequence>1</sequence>
<lineOfBusiness>BUS</lineOfBusiness>
<revenueMgmtSystem>PSRM</revenueMgmtSystem>
<enrollmentId>1aa7720f134c42c98ce9aac9c0a8af3b</enrollmentId>
<status>A</status>
<accessType>TAXROLE</accessType>
<key1>
<name>PER_ID</name>
<value>5786245203</value>
</key1>
<key2>
<name>TAX_ROLE_ID</name>
<value>2057941062</value>
</key2>
</accessEntity>
<accessEntity>
<sequence>2</sequence>
<lineOfBusiness>BUS</lineOfBusiness>
<revenueMgmtSystem>PSRM</revenueMgmtSystem>
<enrollmentId>1aa7720f134c42c98ce9aac9c0a8af3b</enrollmentId>
<status>A</status>
<accessType>TAXROLE</accessType>
<key1>
<name>PER_ID</name>
<value>5786245203</value>
</key1>
<key2>
<name>TAX_ROLE_ID</name>
<value>12311876678</value>
</key2>
</accessEntity>
</userAccess>
</TSEnrollmentServiceRequest>
```

CreateEnrollment Response (Integration Layer to Self Service)

This example shows a final response message sent from the integration layer to the self service application.

```
<ns0:TSEnrollmentServiceRequest xmlns:params="http://schemas.oracle.com/service/bpel/common"
xmlns:ns0="http://oracle.com/TSEnrollmentServiceRequest.xsd" dateTimeTagFormat=""
xmlns="http://oracle.com/TSEnrollmentServiceRequest.xsd">
<ns0:head>
<ns0:key1/>
<ns0:key2/>
<ns0:key3/>
<ns0:key4/>
<ns0:key5/>
<ns0:key6/>
<ns0:key7/>
<ns0:key8/>
<ns0:key9/>
<ns0:key10/>
<ns0:webUserId>USER01</ns0:webUserId>
<ns0:webUserName>John Doe</ns0:webUserName>
<ns0:ipAddress>10.154.138.130</ns0:ipAddress>
</ns0:head>
<ns0:errorMessage>
<ns0:messageTxtOvrd/>
</ns0:errorMessage>
```

```
<ns0:confirmationData>
<ns0:confirmationId>CEID15410000</ns0:confirmationId>
<ns0:header>
<ns0:messageCategory>11126</ns0:messageCategory>
<ns0:messageNumber>30101</ns0:messageNumber>
<ns0:messageTxtOvrd/>
</ns0:header>
<ns0:details>
<ns0:sequence>1</ns0:sequence>
<ns0:messageCategory>11126</ns0:messageCategory>
<ns0:messageNumber>30102</ns0:messageNumber>
<ns0:messageTxtOvrd/>
</ns0:details>
</ns0:confirmationData>
<ns0:mainData>
<ns0:serviceRequestType>ENROLL_BUSINESS</ns0:serviceRequestType>
<ns0:responseMode>SYNCH</ns0:responseMode>
<ns0:serviceRequestData>
<ns0:lineOfBusiness>BUS</ns0:lineOfBusiness>
<ns0:enrollmentId>1aa7720f134c42c98ce9aac9c0a8af3b</ns0:enrollmentId>
<ns0:requestField>
<ns0:sequence>1</ns0:sequence>
<ns0:fieldName>ID_TYPE_BUS</ns0:fieldName>
<ns0:fieldValue>SSN</ns0:fieldValue>
</ns0:requestField>
<ns0:requestField>
<ns0:sequence>2</ns0:sequence>
<ns0:fieldName>ID_VALUE</ns0:fieldName>
<ns0:fieldValue>111-00-6644</ns0:fieldValue>
</ns0:requestField>
<ns0:requestField>
<ns0:sequence>3</ns0:sequence>
<ns0:fieldName>BIRTH_DATE</ns0:fieldName>
<ns0:fieldValue>1966-11-12</ns0:fieldValue>
</ns0:requestField>
</ns0:serviceRequestData>
</ns0:mainData>
</ns0:TSEnrollmentServiceRequest>
```

Enrollment Query Refresh

The examples below illustrate the enrollment query/refresh flow. This flow is initiated when user navigates to My Accounts page.

GetUserEnrollment Request (Self Service to Integration Layer)

The request sent from the self service contains web user id.

```
<ns1:TSGetUserEnrollment xmlns:ns1="http://oracle.com/TSGetUserEnrollment.xsd">
<ns1:head>
<ns1:webUserId>USER01</ns1:webUserId>
<ns1:webUserName>John Doe</ns1:webUserName>
<ns1:emailAddress/>
<ns1:ipAddress>10.154.141.154</ns1:ipAddress>
</ns1:head>
<ns1:mainData>
<ns1:isUserEnrolled>false</ns1:isUserEnrolled>
<ns1:newKeysExist>false</ns1:newKeysExist>
</ns1:mainData>
</ns1:TSGetUserEnrollment>
```

GetUserEnrollment Request (Integration Layer to Revenue Management System)

The integration layer queries user account access data, populates the results on the message and forwards the request to the revenue management system. In this example the user is already enrolled and has access to four individual tax accounts. The request is populated with current user access information.

```
<TSGetUserEnrollment xmlns:params="http://schemas.oracle.com/service/bpel/common"
 xmlns:ns0="http://oracle.com/TSGetUserEnrollment.xsd" dateTimeTagFormat="" xmlns="http://
oracle.com/TSGetUserEnrollment.xsd">
<ns0:head>
<ns0:key1/>
<ns0:key2/>
<ns0:key3/>
<ns0:key4/>
<ns0:key5/>
<ns0:key6/>
<ns0:key7/>
<ns0:key8/>
<ns0:key9/>
<ns0:key10/>
<ns0:webUserId>USER01</ns0:webUserId>
<ns0:webUserName>John Doe</ns0:webUserName>
<ns0:emailAddress/>
<ns0:ipAddress>10.186.253.49</ns0:ipAddress>
</ns0:head>
<ns0:errorMessage>
<ns0:messageParameters/>
</ns0:errorMessage>
<ns0:confirmationData>
<ns0:header>
<ns0:messageParameters/>
</ns0:header>
</ns0:confirmationData>
<ns0:mainData>
<ns0:enrollmentSummary>
<ns0:enrollmentEvent>
<ns0:enrollmentId>3e40e6c2904849a582a06d5cd7127a77<
/ns0:enrollmentId>
</ns0:enrollmentEvent>
</ns0:enrollmentSummary>
<ns0:userAccess>
<ns0:accessEntity>
<ns0:sequence>1</ns0:sequence>
<ns0:lineOfBusiness>IND</ns0:lineOfBusiness>
<ns0:revenueMgmtSystem>ETPM</ns0:revenueMgmtSystem>
<ns0:enrollmentId>3e40e6c2904849a582a06d5cd7127a77<
/ns0:enrollmentId>
<ns0:accessType>TAXROLE</ns0:accessType>
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>6638893506</ns0:value>
</ns0:key1>
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>9037315407</ns0:value>
</ns0:key2>
<ns0:status>A</ns0:status>
</ns0:accessEntity>
<ns0:accessEntity>
<ns0:sequence>2 </ns0:sequence>
<ns0:lineOfBusiness>IND</ns0:lineOfBusiness>
 <ns0:revenueMgmtSystem>PSRM</ns0:revenueMgmtSystem>
 <ns0:enrollmentId>3e40e6c2904849a582a06d5cd7127a77</ns0:enrollmentId>
<ns0:accessType>TAXROLE</ns0:accessType>
```

```
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>6638893506</ns0:value>
</ns0:key1>
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>9037315434</ns0:value>
</ns0:key2>
<ns0:status>A</ns0:status>
</ns0:accessEntity>
<ns0:accessEntity>
<ns0:sequence>3</ns0:sequence>
<ns0:lineOfBusiness>IND</ns0:lineOfBusiness>
 <ns0:revenueMgmtSystem>PSRM</ns0:revenueMgmtSystem>
 <ns0:enrollmentId>3e40e6c2904849a582a06d5cd7127a77</ns0:enrollmentId>
<ns0:accessType>TAXROLE</ns0:accessType>
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>6638893506</ns0:value>
</ns0:key1>
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>9037315640</ns0:value>
</ns0:key2>
<ns0:status>A</ns0:status>
</ns0:accessEntity>
<ns0:accessEntity>
<ns0:sequence>4</ns0:sequence>
<ns0:lineOfBusiness>IND</ns0:lineOfBusiness>
<ns0:revenueMgmtSystem> PSRM</ns0:revenueMgmtSystem>
 <ns0:enrollmentId>3e40e6c2904849a582a06d5cd7127a77</ns0:enrollmentId>
<ns0:accessType>TAXROLE</ns0:accessType>
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>6638893506</ns0:value>
</ns0:key1>
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>9037315933</ns0:value>
</ns0:key2>
<ns0:status>A</ns0:status>
</ns0:accessEntity>
</ns0:userAccess>
</ns0:mainData>
</TSGetUserEnrollment>
```

GetUserEnrollment Response (Revenue Management System to Integration Layer)

The response from the revenue management system is received in the integration layer. In this example the response indicates that user is already enrolled and that new tax accounts were identified. The SOA composite adds new entries to the user account access store and then removes tax account identifiers from the message.

```
<TSGetUserEnrollment xmlns="http://oracle.com/TSGetUserEnrollment.xsd" dateTimeTagFormat="">
<head>
<key1/>
<key2/>
<key3/>
<key4/>
<key5/>
<key6/>
<key7/>
<key8/>
<key9/>
<key10/>
<webUserId>USER01</webUserId>
```

```
<webUserName>John Doe</webUserName>
<emailAddress/>
<ipAddress>10.186.253.49</ipAddress>
</head>
<errorMessage>
<messageParameters/>
</errorMessage>
<confirmationData>
<header>
<messageCategory/>
<messageNumber/>
<messageParameters/>
</header>
</confirmationData>
<mainData>
<isUserEnrolled>true</isUserEnrolled>
<newKeysExist>true</newKeysExist>
<enrollmentSummary>
<enrollmentEvent>
 <enrollmentId>3e40e6c2904849a582a06d5cd7127a77</enrollmentId>
</enrollmentEvent>
</enrollmentSummary>
<userAccess>
<accessEntity>
<sequence>1</sequence>
<lineOfBusiness>IND</lineOfBusiness>
<revenueMgmtSystem>PSRM</revenueMgmtSystem>
 <enrollmentId>3e40e6c2904849a582a06d5cd7127a77</enrollmentId>
<status>A</status>
<accessType>TAXROLE</accessType>
<key1>
<name>PER_ID</name>
<value>6638893506</value>
</key1>
<key2>
<name>TAX_ROLE_ID</name>
<value>9037315275</value>
</key2>
<key3/>
<key4/>
<key5/>
<key6/>
<key7/>
<key8/>
<key9/>
<key10/>
</accessEntity>
<accessEntity>
<sequence>2</sequence>
<lineOfBusiness>IND</lineOfBusiness>
<revenueMgmtSystem>PSRM</revenueMgmtSystem>
 <enrollmentId>3e40e6c2904849a582a06d5cd7127a77</enrollmentId>
<status>A</status>
<accessType>TAXROLE</accessType>
<key1>
<name>PER_ID</name>
<value>6638893506</value>
</key1>
<key2>
<name>TAX_ROLE_ID</name>
<value>9037315639</value>
</key2>
<key3/>
<key4/>
<key5/>
<key6/>
<key7/>
<key8/>
```

```
<key9/>
<key10/>
</accessEntity>
</userAccess>
</mainData>
</TSGetUserEnrollment>
```

GetUserEnrollment Response (Integration Layer to Self Service)

This sample shows the response that is sent from integration layer to the self service application. This final response only contains the list of Line of Business-es that user is enrolled into.

```
<TSGetUserEnrollment xmlns:params="http://schemas.oracle.com/service/bpel/common"
 xmlns:ns0="http://oracle.com/TSGetUserEnrollment.xsd" dateTimeTagFormat="" xmlns="http://
oracle.com/TSGetUserEnrollment.xsd">
<ns0:head>
<ns0:key1/>
<ns0:key2/>
<ns0:key3/>
<ns0:key4/>
<ns0:key5/>
<ns0:key6/>
<ns0:key7/>
<ns0:key8/>
<ns0:key9/>
<ns0:key10/>
<ns0:webUserId>USER01</ns0:webUserId>
<ns0:webUserName>John Doe</ns0:webUserName>
<ns0:emailAddress/>
<ns0:ipAddress>10.186.253.49</ns0:ipAddress>
</ns0:head>
<ns0:errorMessage>
<ns0:messageTxtOvrd/>
</ns0:errorMessage>
<ns0:confirmationData>
<ns0:header>
<ns0:messageCategory>11126</ns0:messageCategory>
<ns0:messageNumber>10004</ns0:messageNumber>
<ns0:messageTxtOvrd/>
</ns0:header>
</ns0:confirmationData>
<ns0:mainData>
<ns0:isUserEnrolled>true</ns0:isUserEnrolled>
<ns0:newKeysExist>true</ns0:newKeysExist>
<ns0:enrollmentSummary>
<ns0:enrollmentEvent>
<ns0:lineOfBusiness>IND</ns0:lineOfBusiness>
</ns0:enrollmentEvent>
</ns0:enrollmentSummary>
</ns0:mainData>
</TSGetUserEnrollment>
```

Enrollment Summary

The examples below illustrate the enrollment summary inquiry request. This flow immediately follows Enrollment Query/ Refresh.

GetEnrollmentSummary Request (Self Service to Integration Layer)

The request sent from the self service contains the web user ID.

```
<ns1:TSGetEnrollmentSummary xmlns:ns1="http://oracle.com/TSGetEnrollmentSummary.xsd">
```

```
<ns1:head>
<ns1:webUserId>USER02</ns1:webUserId>
<ns1:webUserName>Mary Doe</ns1:webUserName>
<ns1:emailAddress/>
<ns1:ipAddress>10.154.153.69</ns1:ipAddress>
</ns1:head>
</ns1:TSGetEnrollmentSummary>
```

GetEnrollmentSummary Request (Integration Layer to Revenue Management System)

The integration layer queries user account access data, populates the results on the message and forwards the request to the revenue management system. In this example the user is already enrolled and has access to individual and business tax accounts. The request is populated with current user access information.

```
<TSGetEnrollmentSummary xmlns:ns0="http://oracle.com/TSGetEnrollmentSummary.xsd"
 dateTimeTagFormat="" xmlns="http://oracle.com/TSGetEnrollmentSummary.xsd">
<ns0:head>
<ns0:key1/>
<ns0:key2/>
<ns0:key3/>
<ns0:key4/>
<ns0:key5/>
<ns0:key6/>
<ns0:key7/>
<ns0:key8/>
<ns0:key9/>
<ns0:key10/>
<ns0:webUserId>USER02</ns0:webUserId>
<ns0:webUserName>Mary Doe</ns0:webUserName>
<ns0:emailAddress/>
<ns0:ipAddress>10.148.114.176</ns0:ipAddress>
</ns0:head>
<ns0:errorMessage>
<ns0:messageParameters/>
</ns0:errorMessage>
<ns0:confirmationData>
<ns0:header>
<ns0:messageParameters/>
</ns0:header>
</ns0:confirmationData>
<ns0:mainData>
<ns0:userAccess>
<ns0:accessEntity>
<ns0:sequence>1</ns0:sequence>
<ns0:lineOfBusiness>IND</ns0:lineOfBusiness>
 <ns0:revenueMgmtSystem>PSRM</ns0:revenueMgmtSystem>
 <ns0:enrollmentId>a8f2a4f83f37474886458731acf13540</ns0:enrollmentId>
<ns0:accessType>TAXROLE</ns0:accessType>
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>6238276573</ns0:value>
</ns0:key1>
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>7255030361</ns0:value>
</ns0:key2>
<ns0:status>C1AP</ns0:status>
</ns0:accessEntity>
<ns0:accessEntity>
<ns0:sequence>2</ns0:sequence>
<ns0:lineOfBusiness>IND</ns0:lineOfBusiness>
 <ns0:revenueMgmtSystem>PSRM</ns0:revenueMgmtSystem>
 <ns0:enrollmentId>a8f2a4f83f37474886458731acf13540</ns0:enrollmentId>
<ns0:accessType>C1TR</ns0:accessType>
<ns0:key1>
```

```
<ns0:name>PER_ID</ns0:name>
<ns0:value>6238276573</ns0:value>
</ns0:key1>
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>7255030624</ns0:value>
</ns0:key2>
<ns0:status>A</ns0:status>
</ns0:accessEntity>
<ns0:accessEntity>
<ns0:sequence>3</ns0:sequence>
<ns0:lineOfBusiness>BUS</ns0:lineOfBusiness>
 <ns0:revenueMgmtSystem>PSRM</ns0:revenueMgmtSystem>
 <ns0:enrollmentId>f522c4e613384fcc93a774f4046912ac</ns0:enrollmentId>
<ns0:accessType>C1TR</ns0:accessType>
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>7801557905</ns0:value>
</ns0:key1>
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>8326266070</ns0:value>
</ns0:key2>
<ns0:status>A</ns0:status>
</ns0:accessEntity>
<ns0:accessEntity>
<ns0:sequence>4</ns0:sequence>
<ns0:lineOfBusiness>BUS</ns0:lineOfBusiness>
 <ns0:revenueMgmtSystem>PSRM</ns0:revenueMgmtSystem>
 <ns0:enrollmentId>f522c4e613384fcc93a774f4046912ac</ns0:enrollmentId>
<ns0:accessType>TAXROLE</ns0:accessType>
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>7801557905</ns0:value>
</ns0:key1>
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>8326266509</ns0:value>
</ns0:key2>
<ns0:status>A</ns0:status>
</ns0:accessEntity>
<ns0:accessEntity>
<ns0:sequence>5</ns0:sequence>
<ns0:lineOfBusiness>BUS</ns0:lineOfBusiness>
 <ns0:revenueMgmtSystem>PSRM</ns0:revenueMgmtSystem>
 <ns0:enrollmentId>f522c4e613384fcc93a774f4046912ac</ns0:enrollmentId>
<ns0:accessType>TAXROLE</ns0:accessType>
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>7801557905</ns0:value>
</ns0:key1>
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>8326266639</ns0:value>
</ns0:key2>
<ns0:status>A</ns0:status>
</ns0:accessEntity>
<ns0:accessEntity>
<ns0:sequence>6</ns0:sequence>
<ns0:lineOfBusiness>BUS</ns0:lineOfBusiness>
 <ns0:revenueMgmtSystem>PSRM</ns0:revenueMgmtSystem>
 <ns0:enrollmentId>f522c4e613384fcc93a774f4046912ac</ns0:enrollmentId>
<ns0:accessType>TAXROLE</ns0:accessType>
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>7801557905</ns0:value>
</ns0:key1>
<ns0:key2>
```

```
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>8326266664</ns0:value>
</ns0:key2>
<ns0:status>A</ns0:status>
</ns0:accessEntity>
</ns0:userAccess>
</ns0:mainData>
</TSGetEnrollmentSummary>
```

GetEnrollmentSummary Response (Revenue Management System to Self Service Portal)

The response contains list of tax accounts. Each entry includes tax account identifiers (access type + keys), line of business and enrollment summary parameters.

```
<TSGetEnrollmentSummary xmlns:params="http://schemas.oracle.com/service/bpel/common"
 xmlns:ns0="http://oracle.com/TSGetEnrollmentSummary.xsd" dateTimeTagFormat=""
 xmlns="http://oracle.com/TSGetEnrollmentSummary.xsd">
<ns0:head>
<ns0:key1/>
<ns0:key2/>
<ns0:key3/>
<ns0:key4/>
<ns0:key5/>
<ns0:key6/>
<ns0:key7/>
<ns0:key8/>
<ns0:key9/>
<ns0:key10/>
<ns0:webUserId>USER02</ns0:webUserId>
<ns0:webUserName>Mary Doe</ns0:webUserName>
<ns0:emailAddress/>
<ns0:ipAddress>10.148.114.176</ns0:ipAddress>
</ns0:head>
<ns0:errorMessage>
<ns0:messageTxtOvrd/>
</ns0:errorMessage>
<ns0:confirmationData>
<ns0:header>
<ns0:messageCategory>11126</ns0:messageCategory>
<ns0:messageNumber>10004</ns0:messageNumber>
<ns0:messageTxtOvrd/>
</ns0:header>
</ns0:confirmationData>
<ns0:mainData>
<ns0:userAccess>
<ns0:accessEntity>
<ns0:sequence>1</ns0:sequence>
<ns0:taxpayerName>Moore, James</ns0:taxpayerName>
 <ns0:isDefaultAccessEntry>true</ns0:isDefaultAccessEntry>
<ns0:lineOfBusiness>IND</ns0:lineOfBusiness>
 <ns0:revenueMgmtSystem>ETPM</ns0:revenueMgmtSystem>
 <ns0:enrollmentId>a8f2a4f83f37474886458731acf13540</ns0:enrollmentId>
<ns0:status>A</ns0:status>
<ns0:accessType>TAXROLE</ns0:accessType>
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>6238276573</ns0:value>
</ns0:key1>
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>7255030361</ns0:value>
</ns0:key2>
<ns0:summaryTitle>
<ns0:parameters>
<ns0:sequence>1</ns0:sequence>
```

```
<ns0:value>ZZINDINCOME</ns0:value>
</ns0:parameters>
</ns0:summaryTitle>
<ns0:summaryDetail>
<ns0:parameters>
<ns0:sequence>1</ns0:sequence>
<ns0:value>2013-03-21</ns0:value>
</ns0:parameters>
<ns0:parameters>
<ns0:sequence>2</ns0:sequence>
</ns0:parameters>
<ns0:parameters>
<ns0:sequence>3</ns0:sequence>
<ns0:value>0.00</ns0:value>
</ns0:parameters>
<ns0:parameters>
<ns0:sequence>4</ns0:sequence>
<ns0:value>2014-04-07</ns0:value>
</ns0:parameters>
</ns0:summaryDetail>
</ns0:accessEntity>
<ns0:accessEntity>
<ns0:sequence>2</ns0:sequence>
<ns0:taxpayerName>Moore, James</ns0:taxpayerName>
 <ns0:isDefaultAccessEntry>false</ns0:isDefaultAccessEntry>
<ns0:lineOfBusiness>IND</ns0:lineOfBusiness>
 <ns0:revenueMgmtSystem>ETPM</ns0:revenueMgmtSystem>
 <ns0:enrollmentId>a8f2a4f83f37474886458731acf13540</ns0:enrollmentId>
<ns0:status>A</ns0:status>
<ns0:accessType>TAXROLE</ns0:accessType>
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>6238276573</ns0:value>
</ns0:key1>
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>7255030624</ns0:value>
</ns0:key2>
<ns0:summaryTitle>
<ns0:parameters>
<ns0:sequence>1</ns0:sequence>
<ns0:value>IND-INCOME</ns0:value>
</ns0:parameters>
</ns0:summaryTitle>
<ns0:summaryDetail>
<ns0:parameters>
<ns0:sequence>1</ns0:sequence>
<ns0:value>2010-01-01</ns0:value>
</ns0:parameters>
<ns0:parameters>
<ns0:sequence>2</ns0:sequence>
</ns0:parameters>
<ns0:parameters>
<ns0:sequence>3</ns0:sequence>
<ns0:value>-2075.00</ns0:value>
</ns0:parameters>
<ns0:parameters>
<ns0:sequence>4</ns0:sequence>
<ns0:value>2014-04-07</ns0:value>
</ns0:parameters>
</ns0:summaryDetail>
</ns0:accessEntity>
<ns0:accessEntity>
<ns0:sequence>3</ns0:sequence>
<ns0:taxpayerName>Cooks For a Cause</ns0:taxpayerName>
 <ns0:isDefaultAccessEntry>true</ns0:isDefaultAccessEntry>
<ns0:lineOfBusiness>BUS</ns0:lineOfBusiness>
<ns0:revenueMgmtSystem>ETPM</ns0:revenueMgmtSystem>
```

```
<ns0:enrollmentId>f522c4e613384fcc93a774f4046912ac</ns0:enrollmentId>
<ns0:status>A</ns0:status>
<ns0:accessType>TAXROLE</ns0:accessType>
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>7801557905</ns0:value>
</ns0:key1>
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>8326266070</ns0:value>
</ns0:key2>
<ns0:summaryTitle>
<ns0:parameters>
<ns0:sequence>1</ns0:sequence>
<ns0:value>SALESTAX</ns0:value>
</ns0:parameters>
</ns0:summaryTitle>
<ns0:summaryDetail>
<ns0:parameters>
<ns0:sequence>1</ns0:sequence>
<ns0:value>1990-11-26</ns0:value>
</ns0:parameters>
<ns0:parameters>
<ns0:sequence>2</ns0:sequence>
<ns0:value>1991-12-31</ns0:value>
</ns0:parameters>
<ns0:parameters>
<ns0:sequence>3</ns0:sequence>
<ns0:value>0.00</ns0:value>
</ns0:parameters>
<ns0:parameters>
<ns0:sequence>4</ns0:sequence>
<ns0:value>2014-04-07</ns0:value>
</ns0:parameters>
<ns0:parameters>
<ns0:sequence>5</ns0:sequence>
<ns0:value>77 Union Streets, San Francisco, CA, 94111</ns0:value>
</ns0:parameters>
</ns0:summaryDetail>
</ns0:accessEntity>
<ns0:accessEntity>
<ns0:sequence>4</ns0:sequence>
<ns0:taxpayerName>Cooks For a Cause</ns0:taxpayerName>
 <ns0:isDefaultAccessEntry>false</ns0:isDefaultAccessEntry>
<ns0:lineOfBusiness>BUS</ns0:lineOfBusiness>
 <ns0:revenueMgmtSystem>PSRM</ns0:revenueMgmtSystem>
 <ns0:enrollmentId>f522c4e613384fcc93a774f4046912ac</ns0:enrollmentId>
<ns0:status>A</ns0:status>
<ns0:accessType>TAXROLE</ns0:accessType>
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>7801557905</ns0:value>
</ns0:key1>
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>8326266509</ns0:value>
</ns0:key2>
<ns0:summaryTitle>
<ns0:parameters>
<ns0:sequence>1</ns0:sequence>
<ns0:value>SALESTAX</ns0:value>
</ns0:parameters>
</ns0:summaryTitle>
<ns0:summaryDetail>
<ns0:parameters>
<ns0:sequence>1</ns0:sequence>
<ns0:value>2001-12-15</ns0:value>
</ns0:parameters>
```

```
<ns0:parameters>
<ns0:sequence>2</ns0:sequence>
</ns0:parameters>
<ns0:parameters>
<ns0:sequence>3</ns0:sequence>
<ns0:value>0.00</ns0:value>
</ns0:parameters>
<ns0:parameters>
<ns0:sequence>4</ns0:sequence>
<ns0:value>2014-04-07</ns0:value>
</ns0:parameters>
<ns0:parameters>
<ns0:sequence>5</ns0:sequence>
<ns0:value>77 Union Streets, San Francisco, CA, 94111</ns0:value>
</ns0:parameters>
</ns0:summaryDetail>
</ns0:accessEntity>
<ns0:accessEntity>
<ns0:sequence>5</ns0:sequence>
<ns0:taxpayerName>Cooks For a Cause</ns0:taxpayerName>
 <ns0:isDefaultAccessEntry>false</ns0:isDefaultAccessEntry>
<ns0:lineOfBusiness>BUS</ns0:lineOfBusiness>
 <ns0:revenueMgmtSystem>PSRM</ns0:revenueMgmtSystem>
 <ns0:enrollmentId>f522c4e613384fcc93a774f4046912ac</ns0:enrollmentId>
<ns0:status>A</ns0:status>
<ns0:accessType>TAXROLE</ns0:accessType>
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>7801557905</ns0:value>
</ns0:key1>
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>8326266639</ns0:value>
</ns0:key2>
<ns0:summaryTitle>
<ns0:parameters>
<ns0:sequence>1</ns0:sequence>
<ns0:value>SALES-USE</ns0:value>
</ns0:parameters>
</ns0:summaryTitle>
<ns0:summaryDetail>
<ns0:parameters>
<ns0:sequence>1</ns0:sequence>
<ns0:value>2008-01-01</ns0:value>
</ns0:parameters>
<ns0:parameters>
<ns0:sequence>2</ns0:sequence>
</ns0:parameters>
<ns0:parameters>
<ns0:sequence>3</ns0:sequence>
<ns0:value>12492.00</ns0:value>
</ns0:parameters>
<ns0:parameters>
<ns0:sequence>4</ns0:sequence>
<ns0:value>2014-04-07</ns0:value>
</ns0:parameters>
<ns0:parameters>
<ns0:sequence>5</ns0:sequence>
<ns0:value>77 Union Streets, San Francisco, CA, 94111</ns0:value>
</ns0:parameters>
</ns0:summaryDetail>
</ns0:accessEntity>
<ns0:accessEntity>
<ns0:sequence>6</ns0:sequence>
<ns0:taxpayerName>Cooks For a Cause</ns0:taxpayerName>
 <ns0:isDefaultAccessEntry>false</ns0:isDefaultAccessEntry>
```

```
<ns0:lineOfBusiness>BUS</ns0:lineOfBusiness>
 <ns0:revenueMgmtSystem>PSRM</ns0:revenueMgmtSystem>
 <ns0:enrollmentId>f522c4e613384fcc93a774f4046912ac</ns0:enrollmentId>
<ns0:status>A</ns0:status>
<ns0:accessType>TAXROLE</ns0:accessType>
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>7801557905</ns0:value>
</ns0:key1>
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>8326266664</ns0:value>
</ns0:key2>
<ns0:summaryTitle>
<ns0:parameters>
<ns0:sequence>1</ns0:sequence>
<ns0:value>CORP-INCOME</ns0:value>
</ns0:parameters>
</ns0:summaryTitle>
<ns0:summaryDetail>
<ns0:parameters>
<ns0:sequence>1</ns0:sequence>
<ns0:value>2008-01-01</ns0:value>
</ns0:parameters>
<ns0:parameters>
<ns0:sequence>2</ns0:sequence>
</ns0:parameters>
<ns0:parameters>
<ns0:sequence>3</ns0:sequence>
<ns0:value>0.00</ns0:value>
</ns0:parameters>
<ns0:parameters>
<ns0:sequence>4</ns0:sequence>
<ns0:value>2014-04-07</ns0:value>
</ns0:parameters>
<ns0:parameters>
<ns0:sequence>5</ns0:sequence>
<ns0:value>77 Union Streets, San Francisco, CA, 94111</ns0:value>
</ns0:parameters>
</ns0:summaryDetail>
</ns0:accessEntity>
</ns0:userAccess>
</ns0:mainData>
</TSGetEnrollmentSummary>
```

Tax Account Alerts

GetTaxAccountAlerts Request

The request contains user id and single tax account's identifiers (access type and access keys).

```
<ns1:TSGetTaxAccountAlerts xmlns:ns1="http://oracle.com/TSGetTaxAccountAlerts.xsd">
<ns1:head>
<ns1:key1>
<ns1:name>PER_ID</ns1:name>
<ns1:value>2885984738</ns1:value>
</ns1:key1>
<ns1:key2>
<ns1:name>TAX_ROLE_ID</ns1:name>
<ns1:value>6294658291</ns1:value>
</ns1:key2>
<ns1:webUserId>USER02</ns1:webUserId>
<ns1:webUserName>Mary Doe</ns1:webUserName>
```

```
<ns1:emailAddress/>
<ns1:ipAddress>10.154.179.54</ns1:ipAddress>
<ns1:accessType>TAXROLE</ns1:accessType>
</ns1:head>
<ns1:lineOfBusiness>BUS</ns1:lineOfBusiness>
</ns1:TSGetTaxAccountAlerts>
```

GetTaxAccountAlerts Response

This sample response contains two alerts.

```
<TSGetTaxAccountAlerts xmlns:params="http://schemas.oracle.com/service/bpel/common"
 xmlns:ns0="http://oracle.com/TSGetTaxAccountAlerts.xsd" dateTimeTagFormat="" xmlns="http://
oracle.com/TSGetTaxAccountAlerts.xsd">
<ns0:head>
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>2885984738</ns0:value>
</ns0:key1>
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>6294658291</ns0:value>
</ns0:key2>
<ns0:key3/>
<ns0:key4/>
<ns0:key5/>
<ns0:key6/>
<ns0:key7/>
<ns0:key8/>
<ns0:key9/>
<ns0:key10/>
<ns0:accessType>TAXROLE</ns0:accessType>
<ns0:webUserId>USER02</ns0:webUserId>
<ns0:webUserName>Mary Doe</ns0:webUserName>
<ns0:emailAddress/>
<ns0:ipAddress>10.154.179.54</ns0:ipAddress>
</ns0:head>
<ns0:lineOfBusiness>BUS</ns0:lineOfBusiness>
<ns0:errorMessage>
<ns0:messageTxtOvrd/>
</ns0:errorMessage>
<ns0:requestData>
<ns0:includeAccountAlerts>true</ns0:includeAccountAlerts>
<ns0:includeGeneralAlerts>true</ns0:includeGeneralAlerts>
</ns0:requestData>
<ns0:responseData>
<ns0:alert>
<ns0:alertType>OVERDUE_BALANCE</ns0:alertType>
<ns0:alertParameters>
<ns0:sequence>1</ns0:sequence>
<ns0:value>18636.61</ns0:value>
</ns0:alertParameters>
</ns0:alert>
<ns0:alert>
<ns0:alertType>TAXPAYER_INFO</ns0:alertType>
</ns0:alert>
</ns0:responseData>
</TSGetTaxAccountAlerts>
```

Tax Account Summary

GetTaxAccountSummary Request

The request contains user id and single tax account's identifiers (access type and access keys).

```
<ns1:TSGetTaxAccountSummary xmlns:ns1="http://oracle.com/TSGetTaxAccountSummary.xsd">
<ns1:head>
<ns1:key1>
<ns1:name>PER_ID</ns1:name>
<ns1:value>2885984738</ns1:value>
</ns1:key1>
<ns1:key2>
<ns1:name>TAX_ROLE_ID</ns1:name>
<ns1:value>6294658291</ns1:value>
</ns1:key2>
<ns1:webUserId>USER02</ns1:webUserId>
<ns1:webUserName>Mary Doe</ns1:webUserName>
<ns1:emailAddress/>
<ns1:ipAddress>10.154.179.54</ns1:ipAddress>
<ns1:accessType>TAXROLE</ns1:accessType>
</ns1:head>
<ns1:lineOfBusiness>BUS</ns1:lineOfBusiness>
</ns1:TSGetTaxAccountSummary>
```

GetTaxAccountSummary Response

```
<TSGetTaxAccountSummary xmlns:params="http://schemas.oracle.com/service/bpel/common"
 xmlns:ns0="http://oracle.com/TSGetTaxAccountSummary.xsd" dateTimeTagFormat=""
 xmlns="http://oracle.com/TSGetTaxAccountSummary.xsd">
<ns0:head>
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>2885984738</ns0:value>
</ns0:key1>
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>6294658291</ns0:value>
</ns0:key2>
<ns0:key3/>
<ns0:key4/>
<ns0:key5/>
<ns0:key6/>
<ns0:key7/>
<ns0:key8/>
<ns0:key9/>
<ns0:key10/>
<ns0:accessType>TAXROLE</ns0:accessType>
<ns0:webUserId>USER02</ns0:webUserId>
<ns0:webUserName>Maty Doe</ns0:webUserName>
<ns0:emailAddress/>
<ns0:ipAddress>10.154.179.54</ns0:ipAddress>
</ns0:head>
<ns0:lineOfBusiness>BUS</ns0:lineOfBusiness>
<ns0:errorMessage>
<ns0:messageTxtOvrd/>
</ns0:errorMessage>
<ns0:responseData>
<ns0:summaryTitleParameters>
<ns0:sequence>1</ns0:sequence>
<ns0:value>West Of Soho</ns0:value>
```

```
</ns0:summaryTitleParameters>
<ns0:summaryTitleParameters>
<ns0:sequence>2</ns0:sequence>
<ns0:value>SALES-USE</ns0:value>
</ns0:summaryTitleParameters>
<ns0:summaryTextParameters>
<ns0:sequence>1</ns0:sequence>
<ns0:value>18636.61</ns0:value>
</ns0:summaryTextParameters>
<ns0:summaryTextParameters>
<ns0:sequence>2</ns0:sequence>
</ns0:summaryTextParameters>
<ns0:summaryTextParameters>
<ns0:sequence>3</ns0:sequence>
</ns0:summaryTextParameters>
<ns0:summaryTextParameters>
<ns0:sequence>4</ns0:sequence>
<ns0:value>2008-04-01</ns0:value>
</ns0:summaryTextParameters>
<ns0:summaryTextParameters>
<ns0:sequence>5</ns0:sequence>
</ns0:summaryTextParameters>
<ns0:currentBalance>18636.61</ns0:currentBalance>
<ns0:taxType>SALES-USE</ns0:taxType>
<ns0:location>
<ns0:effectiveDates/>
<ns0:address>
<ns0:name>West Of Soho</ns0:name>
<ns0:country>USA</ns0:country>
<ns0:address1>304 South Str</ns0:address1>
<ns0:county>Morris</ns0:county>
<ns0:city>Morristown</ns0:city>
<ns0:state>NJ</ns0:state>
<ns0:postal>07960</ns0:postal>
<ns0:inCityLimit>true</ns0:inCityLimit>
</ns0:address>
</ns0:location>
</ns0:responseData>
</TSGetTaxAccountSummary>
```

Tax Account Payments History

GetPaymentHistory Request

The request contains user id and single tax account's identifiers (access type and access keys).

```
<ns1:TSGetPaymentHistory xmlns:ns1="http://oracle.com/TSGetPaymentHistory.xsd">
<ns1:head>
<ns1:key1>
<ns1:name>PER_ID</ns1:name>
<ns1:value>1874847454</ns1:value>
</ns1:key1>
<ns1:key2>
<ns1:name>TAX_ROLE_ID</ns1:name>
<ns1:value>1823125926</ns1:value>
</ns1:key2>
<ns1:webUserId>USER02</ns1:webUserId>
<ns1:webUserName>Mary Doe</ns1:webUserName>
<ns1:emailAddress/>
<ns1:ipAddress>10.154.182.223</ns1:ipAddress>
<ns1:accessType>TAXROLE</ns1:accessType>
</ns1:head>
<ns1:lineOfBusiness>IND</ns1:lineOfBusiness>
```

```
<ns1:requestData/>
</ns1:TSGetPaymentHistory>
```

GetPaymentHistory Response

This sample response contains two payments.

```
<TSGetPaymentHistory xmlns:params="http://schemas.oracle.com/service/bpel/common"
 xmlns:ns0="http://oracle.com/TSGetPaymentHistory.xsd" dateTimeTagFormat="" xmlns="http://
oracle.com/TSGetPaymentHistory.xsd">
<ns0:head>
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>1874847454</ns0:value>
</ns0:key1>
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>1823125926</ns0:value>
</ns0:key2>
<ns0:key3/>
<ns0:key4/>
<ns0:key5/>
<ns0:key6/>
<ns0:key7/>
<ns0:key8/>
<ns0:key9/>
<ns0:key10/>
<ns0:accessType>TAXROLE</ns0:accessType>
<ns0:webUserId>USER02</ns0:webUserId>
<ns0:webUserName>Mary Doe</ns0:webUserName>
<ns0:emailAddress/>
<ns0:ipAddress>10.154.182.223</ns0:ipAddress>
</ns0:head>
<ns0:lineOfBusiness>IND</ns0:lineOfBusiness>
<ns0:errorMessage>
<ns0:messageTxtOvrd/>
</ns0:errorMessage>
<ns0:requestData/>
<ns0:responseData>
<ns0:payment>
<ns0:paymentId>182312514618</ns0:paymentId>
<ns0:amount>22.00</ns0:amount>
<ns0:date>2014-01-31</ns0:date>
<ns0:currency>USD</ns0:currency>
<ns0:status>COMPLETE</ns0:status>
<ns0:paymentType>CHECKING</ns0:paymentType>
<ns0:confirmationId>PTID16500000</ns0:confirmationId>
</ns0:payment>
<ns0:payment>
<ns0:paymentId>182312551858</ns0:paymentId>
<ns0:amount>20.00</ns0:amount>
<ns0:date>2014-01-31</ns0:date>
<ns0:currency>USD</ns0:currency>
<ns0:status>COMPLETE</ns0:status>
<ns0:paymentType>CHECKING</ns0:paymentType>
<ns0:confirmationId>PTID16400000</ns0:confirmationId>
</ns0:payment>
</ns0:responseData>
</TSGetPaymentHistory>
```

Tax Account Filing History

GetFilingHistory Request

The request contains user id and single tax account's identifiers (access type and access keys).

```
<ns1:TSGetFilingHistory xmlns:ns1="http://oracle.com/TSGetFilingHistory.xsd">
<ns1:head>
<ns1:key1>
<ns1:name>PER_ID</ns1:name>
<ns1:value>2885984738</ns1:value>
</ns1:key1>
<ns1:key2>
<ns1:name>TAX_ROLE_ID</ns1:name>
<ns1:value>6294658291</ns1:value>
</ns1:key2>
<ns1:webUserId>USER02</ns1:webUserId>
<ns1:webUserName>Mary Doe</ns1:webUserName>
<ns1:emailAddress/>
<ns1:ipAddress>10.154.179.54</ns1:ipAddress>
<ns1:accessType>TAXROLE</ns1:accessType>
</ns1:head>
<ns1:lineOfBusiness>BUS</ns1:lineOfBusiness>
<ns1:requestData/>
</ns1:TSGetFilingHistory>
```

GetFilingHistory Response

This sample response contains the details for one tax form/filing period.

```
<TSGetFilingHistory xmlns:params="http://schemas.oracle.com/service/bpel/common"
 xmlns:ns0="http://oracle.com/TSGetFilingHistory.xsd" dateTimeTagFormat="" xmlns="http://
oracle.com/TSGetFilingHistory.xsd">
<ns0:head>
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>2885984738</ns0:value>
</ns0:key1>
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>6294658291</ns0:value>
</ns0:key2>
<ns0:key3/>
<ns0:key4/>
<ns0:key5/>
<ns0:key6/>
<ns0:key7/>
<ns0:key8/>
<ns0:key9/>
<ns0:key10/>
<ns0:accessType>TAXROLE</ns0:accessType>
<ns0:webUserId>USER02</ns0:webUserId>
<ns0:webUserName>Mary Doe</ns0:webUserName>
<ns0:emailAddress/>
<ns0:ipAddress>10.154.179.54</ns0:ipAddress>
</ns0:head>
<ns0:lineOfBusiness>BUS</ns0:lineOfBusiness>
<ns0:errorMessage>
<ns0:messageTxtOvrd/>
</ns0:errorMessage>
<ns0:requestData>
```

```
<ns0:taxType/>
</ns0:requestData>
<ns0:responseData>
<ns0:taxForm>
<ns0:formType>SALESDST2008</ns0:formType>
<ns0:formId>928706863748</ns0:formId>
<ns0:filingPeriodStartDate>2010-01-01</ns0:filingPeriodStartDate>
<ns0:filingPeriodEndDate>2010-03-31</ns0:filingPeriodEndDate>
<ns0:taxType>SALES-USE</ns0:taxType>
<ns0:filingStatus>PAYMENTDUE</ns0:filingStatus>
<ns0:confirmationId>TFID50200000</ns0:confirmationId>
<ns0:dueDate>2010-04-15</ns0:dueDate>
<ns0:receivedDate>2014-04-03</ns0:receivedDate>
<ns0:amountDue>18636.61</ns0:amountDue>
<ns0:documentLocatorNumber>TFDLN00000005082</ns0:documentLocatorNumber>
</ns0:taxForm>
</ns0:responseData>
</TSGetFilingHistory>
```

Taxpayer Summary

GetTaxpayerSummary Request

The request contains user id and single tax account's identifiers (access type and access keys).

```
<ns1:TSGetTaxpayerSummary xmlns:ns1="http://oracle.com/TSGetTaxpayerSummary.xsd">
<ns1:head>
<ns1:key1>
<ns1:name>PER_ID</ns1:name>
<ns1:value>7801557905</ns1:value>
</ns1:key1>
<ns1:key2>
<ns1:name>TAX_ROLE_ID</ns1:name>
<ns1:value>8326266070</ns1:value>
</ns1:key2>
<ns1:webUserId>USER01</ns1:webUserId>
<ns1:webUserName>John Doe</ns1:webUserName>
<ns1:emailAddress/>
<ns1:ipAddress>10.186.252.129</ns1:ipAddress>
<ns1:accessType>TAXROLE</ns1:accessType>
</ns1:head>
<ns1:lineOfBusiness>BUS</ns1:lineOfBusiness>
</ns1:TSGetTaxpayerSummary>
```

GetTaxpayerSummary Response

```
<TSGetTaxpayerSummary xmlns:ns0="http://oracle.com/TSGetTaxpayerSummary.xsd"
 dateTimeTagFormat="" xmlns="http://oracle.com/TSGetTaxpayerSummary.xsd">
<ns0:head>
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>7801557905</ns0:value>
</ns0:key1>
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>8326266070</ns0:value>
</ns0:key2>
<ns0:key3/>
<ns0:key4/>
<ns0:key5/>
<ns0:key6/>
<ns0:key7/>
```

```
<ns0:key8/>
<ns0:key9/>
<ns0:key10/>
<ns0:accessType>TAXROLE</ns0:accessType>
<ns0:webUserId>USER01</ns0:webUserId>
<ns0:webUserName>John Doe</ns0:webUserName>
<ns0:emailAddress/>
<ns0:ipAddress>10.186.252.129</ns0:ipAddress>
</ns0:head>
<ns0:lineOfBusiness>BUS</ns0:lineOfBusiness>
<ns0:errorMessage>
<ns0:currency/>
<ns0:messageTxtOvrd/>
</ns0:errorMessage>
<ns0:responseData>
<ns0:name>Cooks For a Cause</ns0:name>
<ns0:taxpayerType>CORPORATION</ns0:taxpayerType>
<ns0:summaryTitleParameters>
<ns0:sequence>1</ns0:sequence>
<ns0:value>Cooks For a Cause</ns0:value>
</ns0:summaryTitleParameters>
<ns0:summaryTitleParameters>
<ns0:sequence>2</ns0:sequence>
<ns0:value>55-8947563</ns0:value>
</ns0:summaryTitleParameters>
<ns0:summaryTextParameters>
<ns0:sequence>1</ns0:sequence>
<ns0:value>CORPORATION</ns0:value>
</ns0:summaryTextParameters>
<ns0:primaryContact>
<ns0:contactType>PRES</ns0:contactType>
<ns0:contactName>Gatmaitan, Francine</ns0:contactName>
<ns0:emailAddress>franG@myMail.com</ns0:emailAddress>
</ns0:primaryContact>
</ns0:responseData>
</TSGetTaxpayerSummary>
```

Taxpayer Contact Info

GetTaxpayerContactInformation Request – Action READ

This sample illustrates the flow invoked on load from Taxpayer Info portal page. The request contains the user ID, the single tax account's identifiers (access type and access keys), and the action.

```
<ns1: TSGetTaxpayerContactInformation xmlns:ns1="http://oracle.com/
 TSGetTaxpayerContactInformation.xsd">
<ns1:head>
<ns0:action>READ</ns0:action>
<ns1:key1>
<ns1:name>PER_ID</ns1:name>
<ns1:value>7801557905</ns1:value>
</ns1:key1>
<ns1:key2>
<ns1:name>TAX_ROLE_ID</ns1:name>
<ns1:value>8326266070</ns1:value>
</ns1:key2>
<ns1:webUserId>USER01</ns1:webUserId>
<ns1:webUserName>John Doe</ns1:webUserName>
<ns1:emailAddress/>
<ns1:ipAddress>10.186.252.129</ns1:ipAddress>
<ns1:accessType>TAXROLE</ns1:accessType>
</ns1:head>
<ns1:lineOfBusiness>BUS</ns1:lineOfBusiness>
```

```
</ns1: TSGetTaxpayerContactInformation>
```

GetTaxpayerContactInformation Response – Action READ

This sample illustrates the flow invoked on load from Taxpayer Info portal page. The response contains the list of phone numbers and the email address.

```
<TSGetTaxpayerContactInformation xmlns:ns0="http://oracle.com/
TSGetTaxpayerContactInformation.xsd" dateTimeTagFormat="" xmlns="http://oracle.com/
TSGetTaxpayerContactInformation.xsd">
<ns0:head>
<ns0:action>READ</ns0:action>
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>5786245203</ns0:value>
</ns0:key1>
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>2057941062</ns0:value>
</ns0:key2>
<ns0:key3/>
<ns0:key4/>
<ns0:key5/>
<ns0:key6/>
<ns0:key7/>
<ns0:key8/>
<ns0:key9/>
<ns0:key10/>
<ns0:accessType>TAXROLE</ns0:accessType>
<ns0:webUserId>USER01</ns0:webUserId>
<ns0:webUserName>John Doe</ns0:webUserName>
<ns0:emailAddress/>
<ns0:ipAddress>10.154.138.130</ns0:ipAddress>
</ns0:head>
<ns0:lineOfBusiness>BUS</ns0:lineOfBusiness>
<ns0:errorMessage>
<ns0:currency/>
<ns0:messageTxtOvrd/>
</ns0:errorMessage>
<ns0:confirmationData>
<ns0:header>
<ns0:messageCategory/>
<ns0:messageTxtOvrd/>
</ns0:header>
</ns0:confirmationData>
<ns0:responseData>
<ns0:phones>
<ns0:sequence>1</ns0:sequence>
<ns0:phoneType>BUSN</ns0:phoneType>
<ns0:phoneNumber>415-884-9888</ns0:phoneNumber>
</ns0:phones>
</ns0:responseData>
</TSGetTaxpayerContactInformation>
```

GetTaxpayerContactInformation Request – Action UPDATE

This sample illustrates the flow invoked when user updates the contact information. The request contains action, user id and single tax account's identifiers (access type and access keys) and also the updated contact information.

```
<ns1:TSGetTaxpayerContactInformation xmlns:ns1="http://oracle.com/
TSGetTaxpayerContactInformation.xsd">
<ns1:head>
<ns1:action>UPDATE</ns1:action>
<ns1:key1>
<ns1:name>PER_ID</ns1:name>
```

```
<ns1:value>5786245203</ns1:value>
</ns1:key1>
<ns1:key2>
<ns1:name>TAX_ROLE_ID</ns1:name>
<ns1:value>2057941062</ns1:value>
</ns1:key2>
<ns1:webUserId>USER01</ns1:webUserId>
<ns1:webUserName>John Doe</ns1:webUserName>
<ns1:emailAddress/>
<ns1:ipAddress>10.154.138.130</ns1:ipAddress>
<ns1:accessType>TAXROLE</ns1:accessType>
</ns1:head>
<ns1:lineOfBusiness>BUS</ns1:lineOfBusiness>
<ns1:responseData>
<ns1:phones>
<ns1:sequence>1</ns1:sequence>
<ns1:phoneType>BUSN</ns1:phoneType>
<ns1:phoneNumber>(415) 884-9888</ns1:phoneNumber>
<ns1:extension>123</ns1:extension>
</ns1:phones>
<ns1:phones>
<ns1:sequence>2</ns1:sequence>
<ns1:phoneType>CELL</ns1:phoneType>
<ns1:phoneNumber>(617) 123-4567</ns1:phoneNumber>
</ns1:phones>
<ns1:email>esmith@jemjewels.com</ns1:email>
</ns1:responseData>
</ns1:TSGetTaxpayerContactInformation>
```

GetTaxpayerContactInformation Response – Action UPDATE

The response contains confirmation details and an updated contact information.

```
<TSGetTaxpayerContactInformation xmlns:ns0="http://oracle.com/
TSGetTaxpayerContactInformation.xsd" dateTimeTagFormat="" xmlns="http://oracle.com/
TSGetTaxpayerContactInformation.xsd">
<ns0:head>
<ns0:action>UPDATE</ns0:action>
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>5786245203</ns0:value>
</ns0:key1>
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>2057941062</ns0:value>
</ns0:key2>
<ns0:key3/>
<ns0:key4/>
<ns0:key5/>
<ns0:key6/>
<ns0:key7/>
<ns0:key8/>
<ns0:key9/>
<ns0:key10/>
<ns0:accessType>TAXROLE</ns0:accessType>
<ns0:webUserId>GUEST</ns0:webUserId>
<ns0:webUserName>GUEST</ns0:webUserName>
<ns0:emailAddress/>
<ns0:ipAddress>10.154.138.130</ns0:ipAddress>
</ns0:head>
<ns0:lineOfBusiness>BUS</ns0:lineOfBusiness>
<ns0:errorMessage>
<ns0:currency/>
<ns0:messageTxtOvrd/>
</ns0:errorMessage>
<ns0:confirmationData>
<ns0:confirmationId>CIID30200000</ns0:confirmationId>
```

```
<ns0:header>
<ns0:messageCategory>11126</ns0:messageCategory>
<ns0:messageNumber>10004</ns0:messageNumber>
<ns0:messageTxtOvrd/>
</ns0:header>
<ns0:details>
<ns0:sequence>1</ns0:sequence>
<ns0:messageCategory>11126</ns0:messageCategory>
<ns0:messageNumber>40004</ns0:messageNumber>
<ns0:messageTxtOvrd/>
</ns0:details>
</ns0:confirmationData>
<ns0:responseData>
<ns0:phones>
<ns0:sequence>1</ns0:sequence>
<ns0:phoneType>BUSN</ns0:phoneType>
<ns0:phoneNumber>(415) 884-9888</ns0:phoneNumber>
<ns0:extension>123</ns0:extension>
</ns0:phones>
<ns0:phones>
<ns0:sequence>2</ns0:sequence>
<ns0:phoneType>CELL</ns0:phoneType>
<ns0:phoneNumber>(617) 123-4567</ns0:phoneNumber>
</ns0:phones>
<ns0:email>esmith@jemjewels.com</ns0:email>
</ns0:responseData>
</TSGetTaxpayerContactInformation>
```

Taxpayer Correspondence Info

GetTaxpayerCorrespondenceInformation Request

The request contains user id and single tax account's identifiers (access type and access keys).

```
<ns1:TSGetTaxpayerCorrespondenceInformation xmlns:ns1="http://oracle.com/
TSGetTaxpayerCorrespondenceInformation.xsd">
<ns1:head>
<ns1:key1>
<ns1:name>PER_ID</ns1:name>
<ns1:value>7801557905</ns1:value>
</ns1:key1>
<ns1:key2>
<ns1:name>TAX_ROLE_ID</ns1:name>
<ns1:value>8326266070</ns1:value>
</ns1:key2>
<ns1:webUserId>USER01</ns1:webUserId>
<ns1:webUserName>John Doe</ns1:webUserName>
<ns1:emailAddress/>
<ns1:ipAddress>10.186.252.129</ns1:ipAddress>
<ns1:accessType>TAXROLE</ns1:accessType>
</ns1:head>
<ns1:lineOfBusiness>BUS</ns1:lineOfBusiness>
</ns1:TSGetTaxpayerCorrespondenceInformation>
```

GetTaxpayerCorrespondenceInformation Response

This sample response contains one address entry.

```
<TSGetTaxpayerCorrespondenceInformation xmlns:ns0="http://oracle.com/
TSGetTaxpayerCorrespondenceInformation.xsd" dateTimeTagFormat="" xmlns="http://oracle.com/
TSGetTaxpayerCorrespondenceInformation.xsd">
<ns0:head>
```

```
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>7801557905</ns0:value>
</ns0:key1>
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>8326266070</ns0:value>
</ns0:key2>
<ns0:key3/>
<ns0:key4/>
<ns0:key5/>
<ns0:key6/>
<ns0:key7/>
<ns0:key8/>
<ns0:key9/>
<ns0:key10/>
<ns0:accessType>TAXROLE</ns0:accessType>
<ns0:webUserId>USER01</ns0:webUserId>
<ns0:webUserName>John Doe</ns0:webUserName>
<ns0:emailAddress/>
<ns0:ipAddress>10.186.252.129</ns0:ipAddress>
</ns0:head>
<ns0:lineOfBusiness>BUS</ns0:lineOfBusiness>
<ns0:errorMessage>
<ns0:currency/>
<ns0:messageTxtOvrd/>
</ns0:errorMessage>
<ns0:responseData>
<ns0:location>
<ns0:sequence>1</ns0:sequence>
<ns0:allowEdit>true</ns0:allowEdit>
<ns0:allowDelete>false</ns0:allowDelete>
<ns0:effectiveDates/>
<ns0:address>
<ns0:country>USA</ns0:country>
<ns0:address1>77 Union Streets</ns0:address1>
<ns0:county>san Francisco</ns0:county>
<ns0:city>San Francisco</ns0:city>
<ns0:state>CA</ns0:state>
<ns0:postal>94111</ns0:postal>
<ns0:inCityLimit>false</ns0:inCityLimit>
</ns0:address>
</ns0:location>
</ns0:responseData>
</TSGetTaxpayerCorrespondenceInformation>
```

Address Maintenance

The address maintenance service is invoked when user adds or edits a single address.

AddressMaintenance Request

This example shows the request sent to the revenue management system. The confirmation id is populated by the integration layer.

```
<TSAddressMaintenance xmlns:ns0="http://oracle.com/TSAddressMaintenance.xsd"
 dateTimeTagFormat="" xmlns="http://oracle.com/TSAddressMaintenance.xsd">
<ns0:head>
<ns0:action>UPDATE</ns0:action>
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>2407057766</ns0:value>
</ns0:key1>
```

```
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>2797085032</ns0:value>
</ns0:key2>
<ns0:key3/>
<ns0:key4/>
<ns0:key5/>
<ns0:key6/>
<ns0:key7/>
<ns0:key8/>
<ns0:key9/>
<ns0:key10/>
<ns0:accessType>TAXROLE</ns0:accessType>
<ns0:webUserId>USER01</ns0:webUserId>
<ns0:webUserName>John Doe</ns0:webUserName>
<ns0:emailAddress/>
<ns0:ipAddress>10.186.252.242</ns0:ipAddress>
</ns0:head>
<ns0:lineOfBusiness>IND</ns0:lineOfBusiness>
<ns0:errorMessage>
<ns0:currency/>
<ns0:messageTxtOvrd/>
</ns0:errorMessage>
<ns0:confirmationData>
<ns0:confirmationId>AMID12200000</ns0:confirmationId>
<ns0:header>
<ns0:messageCategory/>
<ns0:messageNumber/>
<ns0:messageTxtOvrd/>
</ns0:header>
<ns0:details>
<ns0:sequence>1</ns0:sequence>
<ns0:messageCategory>11126</ns0:messageCategory>
<ns0:messageNumber>40003</ns0:messageNumber>
<ns0:messageTxtOvrd/>
</ns0:details>
</ns0:confirmationData>
<ns0:requestData>
<ns0:addressChangeReason>MOVE</ns0:addressChangeReason>
<ns0:location>
<ns0:addressUsage>MAILING</ns0:addressUsage>
<ns0:effectiveDates/>
<ns0:address>
<ns0:country>USA</ns0:country>
<ns0:address1>12 West Lane</ns0:address1>
<ns0:address2/>
<ns0:city>California</ns0:city>
<ns0:state>CA</ns0:state>
<ns0:postal>1234</ns0:postal>
</ns0:address>
</ns0:location>
</ns0:requestData>
</TSAddressMaintenance>
```

AddressMaintenance Response

```
<TSAddressMaintenance xmlns:ns0="http://oracle.com/TSAddressMaintenance.xsd"
 dateTimeTagFormat="" xmlns="http://oracle.com/TSAddressMaintenance.xsd">
<ns0:head>
<ns0:action>UPDATE</ns0:action>
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>2407057766</ns0:value>
</ns0:key1>
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>2797085032</ns0:value>
```

```
</ns0:key2>
<ns0:key3/>
<ns0:key4/>
<ns0:key5/>
<ns0:key6/>
<ns0:key7/>
<ns0:key8/>
<ns0:key9/>
<ns0:key10/>
<ns0:accessType>TAXROLE</ns0:accessType>
<ns0:webUserId>USER01</ns0:webUserId>
<ns0:webUserName>John Doe</ns0:webUserName>
<ns0:emailAddress/>
<ns0:ipAddress>10.186.252.242</ns0:ipAddress>
</ns0:head>
<ns0:lineOfBusiness>IND</ns0:lineOfBusiness>
<ns0:errorMessage>
<ns0:currency/>
<ns0:messageTxtOvrd/>
</ns0:errorMessage>
<ns0:confirmationData>
<ns0:confirmationId>AMID12200000</ns0:confirmationId>
<ns0:header>
<ns0:messageCategory>11126</ns0:messageCategory>
<ns0:messageNumber>10004</ns0:messageNumber>
<ns0:messageTxtOvrd/>
</ns0:header>
<ns0:details>
<ns0:sequence>1</ns0:sequence>
<ns0:messageCategory>11126</ns0:messageCategory>
<ns0:messageNumber>40003</ns0:messageNumber>
<ns0:messageTxtOvrd/>
</ns0:details>
</ns0:confirmationData>
<ns0:requestData>
<ns0:addressChangeReason>MOVE</ns0:addressChangeReason>
<ns0:location>
<ns0:addressUsage>MAILING</ns0:addressUsage>
<ns0:effectiveDates/>
<ns0:address>
<ns0:country>USA</ns0:country>
<ns0:address1>12 West Lane</ns0:address1>
<ns0:address2/>
<ns0:city>California</ns0:city>
<ns0:state>CA</ns0:state>
<ns0:postal>1234</ns0:postal>
</ns0:address>
</ns0:location>
</ns0:requestData>
</TSAddressMaintenance>
```

Process Tax or Registration Form

The examples in this section illustrate the tax/business registration form process flow.

ProcessTaxForm – Action VALIDATE. Request

This example shows the request submitted with action VALIDATE (applicable for all other actions, except SUBMIT)

.

```
<ns1:TSProcessTaxForm xmlns:ns1="http://oracle.com/TSProcessTaxForm.xsd">
<ns1:head>
<ns1:action>VALIDATE</ns1:action>
```

```
<ns1:key1>
<ns1:name>PER_ID</ns1:name>
<ns1:value>2885984738</ns1:value>
</ns1:key1>
<ns1:key2>
<ns1:name>TAX_ROLE_ID</ns1:name>
<ns1:value>6294658291</ns1:value>
</ns1:key2>
<ns1:accessType>TAXROLE</ns1:accessType>
<ns1:webUserId>USER01</ns1:webUserId>
<ns1:webUserName>John Doe</ns1:webUserName>
<ns1:ipAddress>10.154.153.69</ns1:ipAddress>
</ns1:head>
<ns1:requestStatus/>
<ns1:linkedRequest/>
<ns1:mainData>
<ns1:validationRule>SALES-USE</ns1:validationRule>
<ns1:validationServer>http://tudevlv0220.us.oracle.com:9200/determinations-server</
ns1:validationServer>
<ns1:formType>SALESDST2008</ns1:formType>
<ns1:formCategory>TAXFORM</ns1:formCategory>
<ns1:responseMode>SYNCH</ns1:responseMode>
<ns1:formData>
<ns1:id>SALESDST2008</ns1:id>
<ns1:name>SALESDST2008</ns1:name>
<ns1:section>
<ns1:id>75560</ns1:id>
<ns1:name>taxpayerInfo</ns1:name>
<ns1:sequence>1</ns1:sequence>
<ns1:line>
<ns1:field>
<ns1:id>75561</ns1:id>
<ns1:name>name</ns1:name>
<ns1:dataType>TEXT</ns1:dataType>
<ns1:value>West Coast Office Supplies</ns1:value>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75562</ns1:id>
<ns1:name>idType</ns1:name>
<ns1:dataType>LOOKUP</ns1:dataType>
<ns1:value>EIN</ns1:value>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75563</ns1:id>
<ns1:name>idNumber</ns1:name>
<ns1:dataType>TEXT</ns1:dataType>
<ns1:value>82-0000001</ns1:value>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75565</ns1:id>
<ns1:name>address</ns1:name>
<ns1:dataType>TEXT</ns1:dataType>
<ns1:value>123 South Str</ns1:value>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75566</ns1:id>
<ns1:name>address2</ns1:name>
<ns1:dataType>TEXT</ns1:dataType>
<ns1:value/>
</ns1:field>
```

```
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75567</ns1:id>
<ns1:name>city</ns1:name>
<ns1:dataType>TEXT</ns1:dataType>
<ns1:value>Morristown</ns1:value>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75568</ns1:id>
<ns1:name>state</ns1:name>
<ns1:dataType>TEXT</ns1:dataType>
<ns1:value>New Jersey</ns1:value>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75569</ns1:id>
<ns1:name>zipCode</ns1:name>
<ns1:dataType>TEXT</ns1:dataType>
<ns1:value>07960</ns1:value>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75564</ns1:id>
<ns1:name>country</ns1:name>
<ns1:dataType>TEXT</ns1:dataType>
<ns1:value>Uni</ns1:value>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75570</ns1:id>
<ns1:name>accountNumber</ns1:name>
<ns1:dataType>TEXT</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75571</ns1:id>
<ns1:name>filingStartDate</ns1:name>
<ns1:dataType>DATE</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75572</ns1:id>
<ns1:name>filingEndDate</ns1:name>
<ns1:dataType>DATE</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
</ns1:section>
<ns1:section>
<ns1:id>75573</ns1:id>
<ns1:name>lineItems</ns1:name>
<ns1:sequence>1</ns1:sequence>
<ns1:line>
<ns1:field>
<ns1:id>75574</ns1:id>
<ns1:name>totalGrossSales</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value>160000</ns1:value>
```

```
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75575</ns1:id>
<ns1:name>purchasesSubjectToTax</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value>30000</ns1:value>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75576</ns1:id>
<ns1:name>total</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75577</ns1:id>
<ns1:name>totalExemptTransactions</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75578</ns1:id>
<ns1:name>taxableTransactions</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75579</ns1:id>
<ns1:name>taxRate</ns1:name>
<ns1:dataType>NUMBER</ns1:dataType>
<ns1:value>6.25</ns1:value>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75580</ns1:id>
<ns1:name>totalAssessedTaxAmount</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75581</ns1:id>
<ns1:name>transactionsSubjectToLocalTax</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75582</ns1:id>
<ns1:name>localTax</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
```

```
<ns1:id>75583</ns1:id>
<ns1:name>totalTax</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75584</ns1:id>
<ns1:name>firstPrepayment</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75585</ns1:id>
<ns1:name>secondPrepayment</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75586</ns1:id>
<ns1:name>taxPrepayments</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75587</ns1:id>
<ns1:name>remainingTaxDue</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75588</ns1:id>
<ns1:name>penalty</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75589</ns1:id>
<ns1:name>interest</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75590</ns1:id>
<ns1:name>totalAmountOwed</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
</ns1:section>
<ns1:section>
<ns1:id>75597</ns1:id>
<ns1:name>deductExempt</ns1:name>
<ns1:sequence>1</ns1:sequence>
<ns1:line>
```

```
<ns1:field>
<ns1:id>75598</ns1:id>
<ns1:name>salesToOtherRetailers</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value>2000</ns1:value>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75599</ns1:id>
<ns1:name>nonTaxableSales</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value>3000</ns1:value>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75600</ns1:id>
<ns1:name>nonTaxableLabor</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75601</ns1:id>
<ns1:name>salesToGovernment</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75602</ns1:id>
<ns1:name>salesInterstateForeignCommer</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75603</ns1:id>
<ns1:name>salesTaxInGrossSales</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75604</ns1:id>
<ns1:name>lossesTaxableSales</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75605</ns1:id>
<ns1:name>lenderLosses</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75606</ns1:id>
<ns1:name>taxPaidPurchaseResold</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
```

```
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75607</ns1:id>
<ns1:name>returnedTaxableMerchandise</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75608</ns1:id>
<ns1:name>cashDiscountsTaxableSales</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75609</ns1:id>
<ns1:name>totalFullDeductExempt</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
</ns1:section>
<ns1:section>
<ns1:id>75610</ns1:id>
<ns1:name>localTaxCompSchedule</ns1:name>
<ns1:sequence>1</ns1:sequence>
<ns1:line>
<ns1:field>
<ns1:id>75611</ns1:id>
<ns1:name>transactionsSubjectToLocalTax</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value>1000</ns1:value>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75612</ns1:id>
<ns1:name>salesToLocationNotInArea</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75613</ns1:id>
<ns1:name>netAmountSubjectToLocalTax</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75614</ns1:id>
<ns1:name>totalLocalTax</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:table>
<ns1:id>75615</ns1:id>
<ns1:name>compArea</ns1:name>
<ns1:tableRow>
```

```
<ns1:sequence>1</ns1:sequence>
<ns1:field>
<ns1:id>75616</ns1:id>
<ns1:name>localAreaName</ns1:name>
<ns1:dataType>LOOKUP</ns1:dataType>
<ns1:value>CMA2</ns1:value>
</ns1:field>
<ns1:field>
<ns1:id>75617</ns1:id>
<ns1:name>allocatedAmount</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value>1000</ns1:value>
</ns1:field>
<ns1:field>
<ns1:id>75618</ns1:id>
<ns1:name>localAreaTaxRate</ns1:name>
<ns1:dataType>NUMBER</ns1:dataType>
<ns1:value/>
</ns1:field>
<ns1:field>
<ns1:id>75619</ns1:id>
<ns1:name>areaTaxDue</ns1:name>
<ns1:dataType>CURRENCY</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:tableRow>
</ns1:table>
</ns1:section>
<ns1:section>
<ns1:id>75591</ns1:id>
<ns1:name>preparerInformation</ns1:name>
<ns1:sequence>1</ns1:sequence>
<ns1:line>
<ns1:field>
<ns1:id>75592</ns1:id>
<ns1:name>taxpayerTelephoneNumber</ns1:name>
<ns1:dataType>TEXT</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75593</ns1:id>
<ns1:name>preparationDate</ns1:name>
<ns1:dataType>DATE</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75594</ns1:id>
<ns1:name>paidPreparerTelephoneNumber</ns1:name>
<ns1:dataType>TEXT</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75595</ns1:id>
<ns1:name>paidPreparerIdType</ns1:name>
<ns1:dataType>TEXT</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
<ns1:line>
<ns1:field>
<ns1:id>75596</ns1:id>
<ns1:name>paidPreparerIdNumber</ns1:name>
```

```
<ns1:dataType>TEXT</ns1:dataType>
<ns1:value/>
</ns1:field>
</ns1:line>
</ns1:section>
</ns1:formData>
</ns1:mainData>
</ns1:TSProcessTaxForm>
```

ProcessTaxForm – Action SUBMIT. Request

In this example, the request contains tax account identifiers (access keys and access type) and the form data. Integration layer generates the confirmation id and the document locator number and populates it on the request.

```
<TSProcessTaxForm xmlns:params="http://schemas.oracle.com/service/bpel/common"
 xmlns:otss="http://oracle.com/TSProcessTaxForm.xsd" dateTimeTagFormat="" xmlns="http://
oracle.com/TSProcessTaxForm.xsd">
<otss:head>
<otss:action>SUBMIT</otss:action>
<otss:key1>
<otss:name>PER_ID</otss:name>
<otss:value>2885984738</otss:value>
</otss:key1>
<otss:key2>
<otss:name>TAX_ROLE_ID</otss:name>
<otss:value>6294658291</otss:value>
</otss:key2>
<otss:key3/>
<otss:key4/>
<otss:key5/>
<otss:key6/>
<otss:key7/>
<otss:key8/>
<otss:key9/>
<otss:key10/>
<otss:accessType>TAXROLE</otss:accessType>
<otss:webUserId>USER02</otss:webUserId>
<otss:webUserName>Mary Doe</otss:webUserName>
 <otss:emailAddress>myEmail@mail.com</otss:emailAddress>
<otss:ipAddress>10.154.153.69</otss:ipAddress>
</otss:head>
<otss:requestStatus/>
<otss:errorMessage>
<otss:messageTxtOvrd/>
</otss:errorMessage>
<otss:confirmationData>
<otss:confirmationId>TFID51600000</otss:confirmationId>
</otss:confirmationData>
<otss:linkedRequest/>
<otss:mainData>
<otss:formType>SALESDST2008</otss:formType>
<otss:formCategory>TAXFORM</otss:formCategory>
<otss:responseMode>SYNCH</otss:responseMode>
 <otss:documentLocator>TFDLN00000005223</otss:documentLocator>
<otss:formData>
<otss:id>SALESDST2008</otss:id>
<otss:name>SALESDST2008</otss:name>
<otss:section>
<otss:id>75560</otss:id>
<otss:name>taxpayerInfo</otss:name>
<otss:sequence>1</otss:sequence>
<otss:line>
<otss:field>
<otss:id>75561</otss:id>
<otss:name>name</otss:name>
<otss:dataType>TEXT</otss:dataType>
<otss:value>West Of Soho</otss:value>
```

```
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75562</otss:id>
<otss:name>idType</otss:name>
<otss:dataType>LOOKUP</otss:dataType>
<otss:value>EIN</otss:value>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75563</otss:id>
<otss:name>idNumber</otss:name>
<otss:dataType>TEXT</otss:dataType>
<otss:value>78-9999999</otss:value>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75565</otss:id>
<otss:name>address</otss:name>
<otss:dataType>TEXT</otss:dataType>
<otss:value>230 South Str</otss:value>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75566</otss:id>
<otss:name>address2</otss:name>
<otss:dataType>TEXT</otss:dataType>
<otss:value/>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75567</otss:id>
<otss:name>city</otss:name>
<otss:dataType>TEXT</otss:dataType>
<otss:value>Morristown</otss:value>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75568</otss:id>
<otss:name>state</otss:name>
<otss:dataType>TEXT</otss:dataType>
<otss:value>NJ</otss:value>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75569</otss:id>
<otss:name>zipCode</otss:name>
<otss:dataType>TEXT</otss:dataType>
<otss:value>07960</otss:value>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75564</otss:id>
<otss:name>country</otss:name>
<otss:dataType>TEXT</otss:dataType>
<otss:value>USA</otss:value>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
```

```
<otss:id>75570</otss:id>
<otss:name>accountNumber</otss:name>
<otss:dataType>TEXT</otss:dataType>
<otss:value/>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75571</otss:id>
<otss:name>filingStartDate</otss:name>
<otss:dataType>DATE</otss:dataType>
<otss:value>2010-04-01</otss:value>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75572</otss:id>
<otss:name>filingEndDate</otss:name>
<otss:dataType>DATE</otss:dataType>
<otss:value>2010-06-30</otss:value>
</otss:field>
</otss:line>
</otss:section>
<otss:section>
<otss:id>75573</otss:id>
<otss:name>lineItems</otss:name>
<otss:sequence>1</otss:sequence>
<otss:line>
<otss:field>
<otss:id>75574</otss:id>
<otss:name>totalGrossSales</otss:name>
<otss:dataType>CURRENCY</otss:dataType>
<otss:value>160000.0</otss:value>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75575</otss:id>
 <otss:name>purchasesSubjectToTax</otss:name>
<otss:dataType>CURRENCY</otss:dataType>
<otss:value>30000.0</otss:value>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75576</otss:id>
<otss:name>total</otss:name>
<otss:dataType>CURRENCY</otss:dataType>
<otss:value>190000.0</otss:value>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75577</otss:id>
<otss:name>totalExemptTransactions</otss:name>
<otss:dataType>CURRENCY</otss:dataType>
<otss:value>15000.0</otss:value>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75578</otss:id>
<otss:name>taxableTransactions</otss:name>
<otss:dataType>CURRENCY</otss:dataType>
<otss:value>175000.0</otss:value>
</otss:field>
</otss:line>
<otss:line>
```

```
<otss:field>
<otss:id>75579</otss:id>
<otss:name>taxRate</otss:name>
<otss:dataType>NUMBER</otss:dataType>
<otss:value>6.25</otss:value>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75580</otss:id>
 <otss:name>totalAssessedTaxAmount</otss:name>
<otss:dataType>CURRENCY</otss:dataType>
<otss:value>10937.5</otss:value>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75581</otss:id>
 <otss:name>transactionsSubjectToLocalTax</otss:name>
<otss:dataType>CURRENCY</otss:dataType>
<otss:value>3000.0</otss:value>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75582</otss:id>
<otss:name>localTax</otss:name>
<otss:dataType>CURRENCY</otss:dataType>
<otss:value>60.0</otss:value>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75583</otss:id>
<otss:name>totalTax</otss:name>
<otss:dataType>CURRENCY</otss:dataType>
<otss:value>10997.5</otss:value>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75584</otss:id>
<otss:name>firstPrepayment</otss:name>
<otss:dataType>CURRENCY</otss:dataType>
<otss:value>0.0</otss:value>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75585</otss:id>
<otss:name>secondPrepayment</otss:name>
<otss:dataType>CURRENCY</otss:dataType>
<otss:value>0.0</otss:value>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75586</otss:id>
<otss:name>taxPrepayments</otss:name>
<otss:dataType>CURRENCY</otss:dataType>
<otss:value>0.0</otss:value>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75587</otss:id>
<otss:name>remainingTaxDue</otss:name>
<otss:dataType>CURRENCY</otss:dataType>
```

```
<otss:value>10997.5</otss:value>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75588</otss:id>
<otss:name>penalty</otss:name>
<otss:dataType>CURRENCY</otss:dataType>
<otss:value>0.0</otss:value>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75589</otss:id>
<otss:name>interest</otss:name>
<otss:dataType>CURRENCY</otss:dataType>
<otss:value>0.0</otss:value>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75590</otss:id>
<otss:name>totalAmountOwed</otss:name>
<otss:dataType>CURRENCY</otss:dataType>
<otss:value>10997.5</otss:value>
</otss:field>
</otss:line>
</otss:section>
<otss:section>
<otss:id>75610</otss:id>
<otss:name>localTaxCompSchedule</otss:name>
<otss:sequence>1</otss:sequence>
<otss:line>
<otss:field>
<otss:id>75611</otss:id>
<otss:name>transactionsSubjectToLocalTax</otss:name>
<otss:dataType>CURRENCY</otss:dataType>
<otss:value>3000.0</otss:value>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75612</otss:id>
 <otss:name>salesToLocationNotInArea</otss:name>
<otss:dataType>CURRENCY</otss:dataType>
<otss:value>0.0</otss:value>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75613</otss:id>
 <otss:name>netAmountSubjectToLocalTax</otss:name>
<otss:dataType>CURRENCY</otss:dataType>
<otss:value>3000.0</otss:value>
</otss:field>
</otss:line>
<otss:line>
<otss:field>
<otss:id>75614</otss:id>
<otss:name>totalLocalTax</otss:name>
<otss:dataType>CURRENCY</otss:dataType>
<otss:value>60.0</otss:value>
</otss:field>
</otss:line>
<otss:table>
<otss:id>75615</otss:id>
<otss:name>compArea</otss:name>
<otss:tableRow>
```

```
<otss:sequence>1</otss:sequence>
<otss:field>
<otss:id>75616</otss:id>
<otss:name>localAreaName</otss:name>
<otss:dataType>LOOKUP</otss:dataType>
<otss:value>CMA2</otss:value>
</otss:field>
<otss:field>
<otss:id>75617</otss:id>
<otss:name>allocatedAmount</otss:name>
<otss:dataType>CURRENCY</otss:dataType>
<otss:value>3000.0</otss:value>
</otss:field>
<otss:field>
<otss:id>75618</otss:id>
<otss:name>localAreaTaxRate</otss:name>
<otss:dataType>NUMBER</otss:dataType>
<otss:value>0.02</otss:value>
</otss:field>
<otss:field>
<otss:id>75619</otss:id>
<otss:name>areaTaxDue</otss:name>
<otss:dataType>CURRENCY</otss:dataType>
<otss:value>60.0</otss:value>
</otss:field>
</otss:tableRow>
</otss:table>
</otss:section>
</otss:formData>
<otss:exceptions/>
</otss:mainData>
</TSProcessTaxForm>
```

ProcessTaxForm – Action VALIDATE. Response from Form Validation Service in BPEL

The integration layer evaluates request contents and invokes Form Validation service if the validation rule name is populated on the request. This sample illustrates the scenario where exceptions were returned by the validation engine (in this case, OPA rulebase).

```
<FormValidation xmlns:typ="http://oracle.com/determinations/server/10.4/rulebase/assess/
types" xmlns:i18n="http://www.w3.org/2005/09/ws-i18n" xmlns="http://xmlns.oracle.com/OTSS/
OTSSFormValidationService/OTSSFormValidationBPELProcess">
<ots:mainData xmlns:ots="http://xmlns.oracle.com/OTSS/OTSSFormValidationService/
OTSSFormValidationBPELProcess">
<ots:formType>SALESDST2008</ots:formType>
<ots:formData>
<ots:name>SALESDST2008</ots:name>
<ots:id>SALESDST2008</ots:id>
<ots:section>
<ots:id>75560</ots:id>
<ots:name>taxpayerInfo</ots:name>
<ots:sequence>1</ots:sequence>
<ots:line>
<ots:field>
<ots:id>75570</ots:id>
<ots:name>accountNumber</ots:name>
<ots:value/>
</ots:field>
</ots:line>...

[……form data ……]

...<ots:line>
<ots:field>
<ots:id>75609</ots:id>
```

```
<ots:name>totalFullDeductExempt</ots:name>
<ots:value>5000.0</ots:value>
</ots:field>
</ots:line>
</ots:section>
</ots:formData>
<ots:exceptions>
<ots:exception>
<ots:sequence>4</ots:sequence>
<ots:elementReference>75572-1</ots:elementReference>
<ots:errorMessage>
<ots:messageTxtOvrd><b> Filing period end date </b> must be provided</ots:messageTxtOvrd>
</ots:errorMessage>
</ots:exception>
<ots:exception>
<ots:sequence>5</ots:sequence>
<ots:elementReference>75571-1</ots:elementReference>
<ots:errorMessage>
<ots:messageTxtOvrd><b> Filing period start date </b> must be provided</ots:messageTxtOvrd>
</ots:errorMessage>
</ots:exception>
</ots:exceptions>
</ots:mainData>
</FormValidation>
```

ProcessTaxForm – Action VALIDATE. Response

This is a sample response received after invoking action VALIDATE. Similar response is expected from all actions except SUBMIT. Note that the form data validation issues are captured on exceptions collection.

```
<TSProcessTaxForm xmlns:otss="http://oracle.com/TSProcessTaxForm.xsd" dateTimeTagFormat=""
 xmlns="http://oracle.com/TSProcessTaxForm.xsd">
<otss:head>
<otss:action>VALIDATE</otss:action>
<otss:key1>
<otss:name>PER_ID</otss:name>
<otss:value>6638893506</otss:value>
</otss:key1>
<otss:key2>
<otss:name>TAX_ROLE_ID</otss:name>
<otss:value>9037315275</otss:value>
</otss:key2>
<otss:key3/>
<otss:key4/>
<otss:key5/>
<otss:key6/>
<otss:key7/>
<otss:key8/>
<otss:key9/>
<otss:key10/>
<otss:accessType>TAXROLE</otss:accessType>
<otss:webUserId>USER01</otss:webUserId>
<otss:webUserName>John Doe</otss:webUserName>
<otss:ipAddress>10.186.253.49</otss:ipAddress>
</otss:head>
<otss:requestStatus/>
<otss:errorMessage>
<otss:messageParameters/>
</otss:errorMessage>
<otss:confirmationData>
<otss:header>
<otss:messageParameters/>
</otss:header>
</otss:confirmationData>
<otss:linkedRequest/>
<otss:mainData>...
```

```
[……form data ……]

...</otss:line>
</otss:section>
</otss:formData>
<otss:exceptions>
<otss:exception>
<otss:sequence>4</otss:sequence>
<otss:elementReference>75572-1</otss:elementReference>
<otss:errorMessage>
<otss:messageParameters/>
<otss:messageTxtOvrd>Filing period end date must be provided</otss:messageTxtOvrd>
</otss:errorMessage>
</otss:exception>
<otss:exception>
<otss:sequence>5</otss:sequence>
<otss:elementReference>75571-1</otss:elementReference>
<otss:errorMessage>
<otss:messageParameters/>
<otss:messageTxtOvrd>Filing period start date must be provided</otss:messageTxtOvrd>
</otss:errorMessage>
</otss:exception>
</otss:exceptions>
</otss:mainData>
</TSProcessTaxForm>
```

ProcessTaxForm – Action SUBMIT. Response

The response contains confirmation details. Note that form data is echoed back to the self service application. This data is used if user chooses to print the form.

```
<TSProcessTaxForm xmlns:params="http://schemas.oracle.com/service/bpel/common"
xmlns:otss="http://oracle.com/TSProcessTaxForm.xsd" dateTimeTagFormat=""
xmlns="http://oracle.com/TSProcessTaxForm.xsd">
<otss:head>
<otss:action>SUBMIT</otss:action>
<otss:key1>
<otss:name>PER_ID</otss:name>
<otss:value>2885984738</otss:value>
</otss:key1>
<otss:key2>
<otss:name>TAX_ROLE_ID</otss:name>
<otss:value>6294658291</otss:value>
</otss:key2>
<otss:key3/>
<otss:key4/>
<otss:key5/>
<otss:key6/>
<otss:key7/>
<otss:key8/>
<otss:key9/>
<otss:key10/>
<otss:accessType>TAXROLE</otss:accessType>
<otss:webUserId>USER02</otss:webUserId>
<otss:webUserName>Mary Doe</otss:webUserName>
<otss:emailAddress>myEmail@mail.com</otss:emailAddress>
<otss:ipAddress>10.154.153.69</otss:ipAddress>
</otss:head>
<otss:requestStatus/>
<otss:errorMessage>
<otss:messageTxtOvrd/>
</otss:errorMessage>
<otss:confirmationData>
<otss:confirmationId>TFID51600000</otss:confirmationId>
<otss:header>
<otss:messageCategory>11126</otss:messageCategory>
<otss:messageNumber>10004</otss:messageNumber>
```

```
<otss:messageTxtOvrd/>
</otss:header>
</otss:confirmationData>
<otss:linkedRequest/>
<otss:mainData>
<otss:formType>SALESDST2008</otss:formType>
<otss:formCategory>TAXFORM</otss:formCategory>
<otss:responseMode>SYNCH</otss:responseMode>
<otss:documentLocator>TFDLN00000005223</otss:documentLocator>
<otss:formData>
<otss:id>SALESDST2008</otss:id>
<otss:name>SALESDST2008</otss:name>
<otss:section>
<otss:id>75560</otss:id>
<otss:name>taxpayerInfo</otss:name>
<otss:sequence>1</otss:sequence>
<otss:line>

[……form data ……]

<otss:field>
<otss:id>75619</otss:id>
<otss:name>areaTaxDue</otss:name>
<otss:dataType>CURRENCY</otss:dataType>
<otss:value>60.0</otss:value>
</otss:field>
</otss:tableRow>
</otss:table>
</otss:section>
</otss:formData>
<otss:exceptions/>
</otss:mainData>
</TSProcessTaxForm>
```

Print Form Data

PrintForm Request

The request contains a user ID, form category, and the language currently used on the self service portal.

```
<ns1:TSPrintDocument xmlns:ns1="http://oracle.com/TSPrintDocument.xsd">
<ns1:head>
<ns1:action/>
<ns1:key1>
<ns1:name>PER_ID</ns1:name>
<ns1:value>4121934169</ns1:value>
</ns1:key1>
<ns1:key2>
<ns1:name>TAX_ROLE_ID</ns1:name>
<ns1:value>8690587828</ns1:value>
</ns1:key2>
<ns1:webUserId>USER02</ns1:webUserId>
<ns1:webUserName>Mary Doe</ns1:webUserName>
<ns1:ipAddress>10.154.117.135</ns1:ipAddress>
<ns1:accessType>TAXROLE</ns1:accessType>
</ns1:head>
<ns1:requestStatus/>
<ns1:linkedRequest/>
<ns1:input>
<ns1:documentSource>TAXFORM</ns1:documentSource>
<ns1:documentData>
<documentData>
<formType>ZZ-SS-TEST02</formType>
```

```
<formCategory>TAXFORM</formCategory>
<currency/>
<formTypeDescription>Self Service Test Form 02 - Internet Purchases</formTypeDescription>
<documentLocator/>
<documentLocatorLabel>Document Locator</documentLocatorLabel>
<customProperties/>
<formData>
<id>ZZ-SS-TEST02</id>
<name>Self Service Test Form 02 - Internet Purchases</name>
<section>
<id>63821</id>
<name>identification</name>
<sectionLabel>Taxpayer Identification</sectionLabel>
<sequence>1</sequence>
<line>
<field>
<id>63822</id>
<name>name</name>
<lineLabel>Name</lineLabel>
<dataType>TEXT</dataType>
<value>John Segal</value>
</field>
</line>
<line>
<field>
<id>63823</id>
<name>idType</name>
<lineLabel>ID Type</lineLabel>
<dataType>LOOKUP</dataType>
<value>Social Security Number </value>
</field>
</line>
<line>
<field>
<id>63824</id>
<name>idNumber</name>
<lineLabel>ID Number</lineLabel>
<dataType>TEXT</dataType>
<value>666-77-8888</value>
</field>
</line>
<line>
<field>
<id>63825</id>
<name>address</name>
<lineLabel>Address</lineLabel>
<dataType>TEXT</dataType>
<value>400 Crossing Blvd</value>
</field>
</line>
<line>
<field>
<id>63826</id>
<name>city</name>
<lineLabel>City</lineLabel>
<dataType>TEXT</dataType>
<value>Bridgewater</value>
</field>
</line>
<line>
<field>
<id>63827</id>
<name>state</name>
<lineLabel>State</lineLabel>
<dataType>TEXT</dataType>
<value>NJ</value>
</field>
</line>
```

```
<line>
<field>
<id>63828</id>
<name>postal</name>
<lineLabel>Postal</lineLabel>
<dataType>TEXT</dataType>
<value>08648</value>
</field>
</line>
<line>
<field>
<id>63829</id>
<name>contry</name>
<lineLabel>Country</lineLabel>
<dataType>TEXT</dataType>
<value>USA</value>
</field>
</line>
<line>
<field>
<id>63830</id>
<name>fileStartDate</name>
<lineLabel>Filing Period Start Date</lineLabel>
<dataType>DATE</dataType>
<value>06/10/2014</value>
</field>
</line>
<line>
<field>
<id>63831</id>
<name>fileEndDate</name>
<lineLabel>Filing Period End Date</lineLabel>
<dataType>DATE</dataType>
<value>06/17/2014</value>
</field>
</line>
</section>
<section>
<id>63832</id>
<name>financialData</name>
<sectionLabel>Financial Data</sectionLabel>
<sequence>1</sequence>
<line>
<field>
<id>63833</id>
<name>stateInternetPurchases</name>
<lineLabel>Total Internet Purchases in State</lineLabel>
<dataType>CURRENCY</dataType>
<value>1</value>
</field>
</line>
<line>
<field>
<id>63834</id>
<name>outInternetPurchases</name>
<lineLabel>Internet Purchases out of State</lineLabel>
<dataType>CURRENCY</dataType>
<value>1</value>
</field>
</line>
<line>
<field>
<id>63835</id>
<name>totalPurchases</name>
<lineLabel>Total Internet Purchases</lineLabel>
<dataType>CURRENCY</dataType>
<value>2</value>
</field>
```

```
</line>
<line>
<field>
<id>63836</id>
<name>exemptPurchases</name>
<lineLabel>Purchases exempt from Tax</lineLabel>
<dataType>CURRENCY</dataType>
<value>1</value>
</field>
</line>
<line>
<field>
<id>63837</id>
<name>totalTaxablePurchases</name>
<lineLabel>Total Taxable Purchases</lineLabel>
<dataType>CURRENCY</dataType>
<value>1</value>
</field>
</line>
<line>
<field>
<id>63838</id>
<name>taxRateType</name>
<lineLabel>Tax Rate Type</lineLabel>
<dataType>LOOKUP</dataType>
<value>Self Determined Rate</value>
</field>
</line>
<line>
<field>
<id>63839</id>
<name>taxRateValue</name>
<lineLabel>Tax Rate Value</lineLabel>
<dataType>NUMBER</dataType>
<value>1000000</value>
</field>
</line>
<line>
<field>
<id>63840</id>
<name>calcTax</name>
<lineLabel>Calculated Tax</lineLabel>
<dataType>CURRENCY</dataType>
<value>1</value>
</field>
</line>
<line>
<field>
<id>63841</id>
<name>prepayAmount</name>
<lineLabel>Prepayment Amount Received</lineLabel>
<dataType>CURRENCY</dataType>
<value>1</value>
</field>
</line>
<line>
<field>
<id>63842</id>
<name>totalTaxDue</name>
<lineLabel>Total Tax Due</lineLabel>
<dataType>CURRENCY</dataType>
<value>1</value>
</field>
</line>
</section>
<section>
<id>63843</id>
<name>dependentsInfo</name>
```

```
<sectionLabel>Dependents Information</sectionLabel>
<sequence>1</sequence>
<table>
<id>63844</id>
<name>dependentTable</name>
<tableLabel>Dependent Table</tableLabel>
<tableRow>
<sequence>1</sequence>
<field>
<id>63845</id>
<name>name</name>
<fieldLabel>Name</fieldLabel>
<dataType>TEXT</dataType>
<value>test</value>
</field>
<field>
<id>63846</id>
<name>age</name>
<fieldLabel>Age</fieldLabel>
<dataType>NUMBER</dataType>
<value>1</value>
</field>
<field>
<id>63847</id>
<name>schoolGrade</name>
<fieldLabel>School Grade Number</fieldLabel>
<dataType>NUMBER</dataType>
<value>1</value>
</field>
</tableRow>
</table>
</section>
</formData>
</documentData>
</ns1:documentData>
</ns1:input>
</ns1:TSPrintDocument>
```

ReportService (BI Publisher) call

The samples illustrate the interaction with document generation flow. The provided solution uses Oracle Business Intelligence Publisher for printable form document creation.

ReportService, operation RunReport Request

The service accepts input data in two forms: raw XML and base64-encoded binary stream. The example below shows raw XML.

The integration layer configuration defines if the data needs to be base64-encoded.

```
<runReport xmlns="http://xmlns.oracle.com/oxp/service/v2">
<reportRequest>
<XDOPropertyList/>
<attributeCalendar/>
<attributeFormat>pdf</attributeFormat>
<attributeLocale/>
<attributeTemplate/>
<attributeTimezone/>
<byPassCache>true</byPassCache>
<dynamicDataSource/>
<flattenXML>false</flattenXML>
<parameterNameValues/>
<reportAbsolutePath>/~weblogic/Drafts/GenericFormData.xdo</reportAbsolutePath>
<reportData/>
<reportOutputPath/>
```

<reportRawData><documentData><formType>SALESDST2008</formType><formCategory>TAXFORM</
formCategory><currency>USD</currency><formTypeDescription>Sales and Use
 (w/ Local Distribution) Tax Form</formTypeDescription><documentLocator/
><documentLocatorLabel>Document Locator</documentLocatorLabel><customProperties/
><formData><id>SALESDST2008</id><name>Sales and Use (w/ Local Distribution) Tax
 Form</name><section><id>75560</id><name>taxpayerInfo</name><sectionLabel>Taxpayer
 Information</sectionLabel><sequence>1</sequence><line><field><id>75561</
id><name>name</name><lineLabel>Name</lineLabel><dataType>TEXT</dataType><value>West
 of Soho</value></field></line><line><field><id>75562</id><name>idType</
name><lineLabel>ID Type</lineLabel><dataType>LOOKUP</dataType><value>Employer
 Identification Number</value></field></line><line><field><id>75563</
id><name>idNumber</name><lineLabel>ID Number</lineLabel><dataType>TEXT</
dataType><value>78-9999999</value></field></line><line><field><id>75565</
id><name>address</name><lineLabel>Address</lineLabel><dataType>TEXT</dataType><value>235
 South Str</value></field></line><line><field><id>75566</id><name>address2</
name><lineLabel>Address 2</lineLabel><dataType>TEXT</dataType><value/></field></
line><line><field><id>75567</id><name>city</name><lineLabel>City</lineLabel><dataType>TEXT</
dataType><value>Morristown</value></field></line><line><field><id>75568</
id><name>state</name><lineLabel>State</lineLabel><dataType>TEXT</dataType><value>New
 Jersey</value></field></line><line><field><id>75569</id><name>zipCode</
name><lineLabel>Postal</lineLabel><dataType>TEXT</dataType><value>07960</value></
field></line><line><field><id>75564</id><name>country</name><lineLabel>Country</
lineLabel><dataType>TEXT</dataType><value>USA</value></field></line><line><field><id>75570</
id><name>accountNumber</name><lineLabel>Account</lineLabel><dataType>TEXT</dataType><value/
></field></line><line><field><id>75571</id><name>filingStartDate</name><lineLabel>Filing
 Period Start Date</lineLabel><dataType>DATE</dataType><value>04/01/2010</value></
field></line><line><field><id>75572</id><name>filingEndDate</name><lineLabel>Filing
 Period End Date</lineLabel><dataType>DATE</dataType><value>06/30/2010</value></field></
line></section><section><id>75573</id><name>lineItems</name><sectionLabel>Line Items</
sectionLabel><sequence>1</sequence><line><field><id>75574</id><name>totalGrossSales</
name><lineLabel>Total Gross Sales</lineLabel><dataType>CURRENCY</dataType><value>180000.0</
value></field></line><line><field><id>75575</id><name>purchasesSubjectToTax</
name><lineLabel>Purchases Subject to Tax </lineLabel><dataType>CURRENCY</
dataType><value>30000.0</value></field></line><line><field><id>75576</id><name>total</
name><lineLabel>Total Assessed Tax Amount</lineLabel><dataType>CURRENCY</
dataType><value>210000.0</value></field></line><line><field><id>75577</
id><name>totalExemptTransactions</name><lineLabel>Total Deductions / Exemptions</
lineLabel><dataType>CURRENCY</dataType><value>7300.0</value></field></
line><line><field><id>75578</id><name>taxableTransactions</name><lineLabel>Transactions
 Subject To Tax</lineLabel><dataType>CURRENCY</dataType><value>202700.0</value></
field></line><line><field><id>75579</id><name>taxRate</name><lineLabel>State
 Tax Rate</lineLabel><dataType>NUMBER</dataType><value>6.25</value></field></
line><line><field><id>75580</id><name>totalAssessedTaxAmount</name><lineLabel>Assessed
 Tax Amount</lineLabel><dataType>CURRENCY</dataType><value>12668.75</value></
field></line><line><field><id>75581</id><name>transactionsSubjectToLocalTax</
name><lineLabel>Transactions Subject To Local Tax</lineLabel><dataType>CURRENCY</
dataType><value>1000.0</value></field></line><line><field><id>75582</id><name>localTax</
name><lineLabel>Local Tax</lineLabel><dataType>CURRENCY</dataType><value>20.0</
value></field></line><line><field><id>75583</id><name>totalTax</name><lineLabel>Total
 Tax</lineLabel><dataType>CURRENCY</dataType><value>12688.75</value></field></
line><line><field><id>75584</id><name>firstPrepayment</name><lineLabel>First
 Prepayment</lineLabel><dataType>CURRENCY</dataType><value>0.0</value></field></
line><line><field><id>75585</id><name>secondPrepayment</name><lineLabel>Second
 Prepayment</lineLabel><dataType>CURRENCY</dataType><value>0.0</value></field></
line><line><field><id>75586</id><name>taxPrepayments</name><lineLabel>Total Tax
 Prepayments</lineLabel><dataType>CURRENCY</dataType><value>0.0</value></field></
line><line><field><id>75587</id><name>remainingTaxDue</name><lineLabel>Remaining
 Tax Due</lineLabel><dataType>CURRENCY</dataType><value>12688.75</value></
field></line><line><field><id>75588</id><name>penalty</name><lineLabel>Penalty</
lineLabel><dataType>CURRENCY</dataType><value>0.0</value></field></
line><line><field><id>75589</id><name>interest</name><lineLabel>Interest</
lineLabel><dataType>CURRENCY</dataType><value>0.0</value></field></
line><line><field><id>75590</id><name>totalAmountOwed</name><lineLabel>Total Amount
 Owed</lineLabel><dataType>CURRENCY</dataType><value>12688.75</value></field></line></
section><section><id>75597</id><name>deductExempt</name><sectionLabel>Schedule A -
 Deductions / Exemptions</sectionLabel><sequence>1</sequence><line><field><id>75598</
id><name>salesToOtherRetailers</name><lineLabel>Sales to Other Retailers</
lineLabel><dataType>CURRENCY</dataType><value>5000.0</value></field></

```
line><line><field><id>75599</id><name>nonTaxableSales</name><lineLabel>Non-Taxable
 Sales of Food Products</lineLabel><dataType>CURRENCY</dataType><value>2300.0</value></
field></line><line><field><id>75600</id><name>nonTaxableLabor</name><lineLabel>Non-
Taxable Labor</lineLabel><dataType>CURRENCY</dataType><value>0.0</value></field></
line><line><field><id>75601</id><name>salesToGovernment</name><lineLabel>Sales to
 Government</lineLabel><dataType>CURRENCY</dataType><value>0.0</value></field></
line><line><field><id>75602</id><name>salesInterstateForeignCommer</name><lineLabel>Sales
 in Interstate Foreign Commerce</lineLabel><dataType>CURRENCY</dataType><value>0.0</value></
field></line><line><field><id>75603</id><name>salesTaxInGrossSales</name><lineLabel>Sales
 Tax in Gross Sales</lineLabel><dataType>CURRENCY</dataType><value>0.0</value></field></
line><line><field><id>75604</id><name>lossesTaxableSales</name><lineLabel>Bad Debt
 Losses on Taxable Sales </lineLabel><dataType>CURRENCY</dataType><value>0.0</value></
field></line><line><field><id>75605</id><name>lenderLosses</name><lineLabel>Bad Debt
 Lender Losses</lineLabel><dataType>CURRENCY</dataType><value>0.0</value></field></
line><line><field><id>75606</id><name>taxPaidPurchaseResold</name><lineLabel>Cost of Tax-
Paid Purchases Resold Prior to Use</lineLabel><dataType>CURRENCY</dataType><value>0.0</
value></field></line><line><field><id>75607</id><name>returnedTaxableMerchandise</
name><lineLabel>Returned Taxable Merchandise</lineLabel><dataType>CURRENCY</
dataType><value>0.0</value></field></line><line><field><id>75608</
id><name>cashDiscountsTaxableSales</name><lineLabel>Cash Discounts on Taxable
 Sales</lineLabel><dataType>CURRENCY</dataType><value>0.0</value></field></
line><line><field><id>75609</id><name>totalFullDeductExempt</name><lineLabel>Total
 Deductions / Exemptions</lineLabel><dataType>CURRENCY</dataType><value>7300.0</
value></field></line></section><section><id>75610</id><name>localTaxCompSchedule</
name><sectionLabel>Schedule B - Local Tax Computation</sectionLabel><sequence>1</
sequence><line><field><id>75611</id><name>transactionsSubjectToLocalTax</
name><lineLabel>Transactions Subject to Local Tax</lineLabel><dataType>CURRENCY</
dataType><value>1000.0</value></field></line><line><field><id>75612</
id><name>salesToLocationNotInArea</name><lineLabel>Sales Delivered to Locations Not
 in Any Area</lineLabel><dataType>CURRENCY</dataType><value>0.0</value></field></
line><line><field><id>75613</id><name>netAmountSubjectToLocalTax</name><lineLabel>Net
 Amount Subject to Local Tax</lineLabel><dataType>CURRENCY</dataType><value>1000.0</
value></field></line><table><id>75615</id><name>compArea</name><tableLabel>Computation
 for Local Area</tableLabel><tableRow><sequence>1</sequence><field><id>75616</
id><name>localAreaName</name><fieldLabel>Local Area</fieldLabel><dataType>LOOKUP</
dataType><value>Chester</value></field><field><id>75617</id><name>allocatedAmount</
name><fieldLabel>Allocated Amount</fieldLabel><dataType>CURRENCY</dataType><value>1000.0</
value></field><field><id>75618</id><name>localAreaTaxRate</name><fieldLabel>Local
 Area Tax Rate</fieldLabel><dataType>NUMBER</dataType><value>0.02</value></
field><field><id>75619</id><name>areaTaxDue</name><fieldLabel>Area Tax Due</
fieldLabel><dataType>CURRENCY</dataType><value>20.0</value></field></tableRow></
table><line><field><id>75614</id><name>totalLocalTax</name><lineLabel>Total
 Local Tax</lineLabel><dataType>CURRENCY</dataType><value>20.0</value></
field></line></section><section><id>75591</id><name>preparerInformation</
name><sectionLabel>Preparer Information</sectionLabel><sequence>1</
sequence><line><field><id>75592</id><name>taxpayerTelephoneNumber</name><lineLabel>Taxpayer
 Telephone Number</lineLabel><dataType>TEXT</dataType><value/></field></
line><line><field><id>75593</id><name>preparationDate</name><lineLabel>Preparation
 Date</lineLabel><dataType>DATE</dataType><value/></field></line><line><field><id>75594</
id><name>paidPreparerTelephoneNumber</name><lineLabel>Paid Preparer Telephone Number</
lineLabel><dataType>TEXT</dataType><value/></field></line><line><field><id>75595</
id><name>paidPreparerIdType</name><lineLabel>Paid Preparer ID Type</
lineLabel><dataType>TEXT</dataType><value/></field></line><line><field><id>75596</
id><name>paidPreparerIdNumber</name><lineLabel>Paid Preparer ID Number</
lineLabel><dataType>TEXT</dataType><value/></field></line></section></formData></
documentData></reportRawData>
<sizeOfDataChunkDownload>-1</sizeOfDataChunkDownload>
</reportRequest>
<userID>WSSUSER</userID>
<password>password</password>
</runReport>
ReportService, operation RunReport Response
<runReportResponse xmlns="http://xmlns.oracle.com/oxp/service/v2"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
<runReportReturn>
<metaDataList xsi:nil="true"/>
<reportBytes>
```

```
…[ report data, base64-encoded binary stream]…
</reportBytes>
<reportContentType>application/pdf</reportContentType>
<reportFileID xsi:nil="true"/>
<reportLocale xsi:nil="true"/>
</runReportReturn>
</runReportResponse>
```

PrintForm Response

T
he response contains base-64 encoded document contents and the MIME type.

```
<TSPrintDocument xmlns:ns0="http://oracle.com/TSPrintDocument.xsd" dateTimeTagFormat=""
xmlns="http://oracle.com/TSPrintDocument.xsd">
<ns0:head>
<ns0:action/>
<ns0:key1>
<ns0:name>PER_ID</ns0:name>
<ns0:value>2885984738</ns0:value>
</ns0:key1>
<ns0:key2>
<ns0:name>TAX_ROLE_ID</ns0:name>
<ns0:value>6294658291</ns0:value>
</ns0:key2>
<ns0:key3/>
<ns0:key4/>
<ns0:key5/>
<ns0:key6/>
<ns0:key7/>
<ns0:key8/>
<ns0:key9/>
<ns0:key10/>
<ns0:accessType>TAXROLE</ns0:accessType>
<ns0:webUserId>SPLVXF</ns0:webUserId>
<ns0:webUserName>SPLVXF</ns0:webUserName>
<ns0:ipAddress>10.154.153.69</ns0:ipAddress>
</ns0:head>
<ns0:requestStatus/>
<ns0:errorMessage>
<ns0:messageParameters/>
</ns0:errorMessage>
<ns0:confirmationData>
<ns0:header>
<ns0:messageParameters/>
</ns0:header>
</ns0:confirmationData>
<ns0:linkedRequest/>
<ns0:input>
<ns0:documentSource>TAXFORM</ns0:documentSource>
<ns0:documentData>
... [……raw XML……]...
</ns0:documentData>
</ns0:input>
<ns0:output>
<ns0:documentContent>
... [……base64-encoded stream……]...
</ns0:documentContent>
<ns0:documentType>application/pdf</ns0:documentType>
</ns0:output>
</TSPrintDocument>
```

Upload File

UploadSupportingDocument Request (Self Service to Integration Layer)

The request originated from the self service application includes uploaded file details, tax account identifiers and user id. This example illustrates file upload with target form's Document Locator Number. The name of the uploaded file is captured in <fileURL> element. File contents are base-64 encoded.

```
<ns1:TSUploadSupportingDocument xmlns:ns1="http://oracle.com/
TSUploadSupportingDocument.xsd">
<ns1:head>
<ns1:key1>
<ns1:name>PER_ID</ns1:name>
<ns1:value>6638893506</ns1:value>
</ns1:key1>
<ns1:key2>
<ns1:name>TAX_ROLE_ID</ns1:name>
<ns1:value>9037315275</ns1:value>
</ns1:key2>
<ns1:accessType>TAXROLE</ns1:accessType>
<ns1:webUserId>USER01</ns1:webUserId>
<ns1:webUserName>John Doe</ns1:webUserName>
<ns1:emailAddress/>
<ns1:ipAddress>10.186.253.49</ns1:ipAddress>
</ns1:head>
<ns1:mainData>
<ns1:documentLocator>TFDLN00000005868</ns1:documentLocator>
<ns1:fileURL>MySample.txt</ns1:fileURL>
<ns1:content>aGVsbG8gd29ybGQ=</ns1:content>
</ns1:mainData>
</ns1:TSUploadSupportingDocument>
```

Interaction with Document Repository (Oracle Web Content Management) (Integration Layer)

The integration layer generates a new unique file name and sends file to the document repository via a web service. The provided solution interacts with the Oracle Web Content Management server.

The response is expected to contain the location of the newly-uploaded file.

Generic Service, Request

```
<GenericRequest webKey="cs" xmlns:params="http://schemas.oracle.com/service/bpel/common"
 xmlns:ns1="http://www.oracle.com/UCM" xmlns="http://www.oracle.com/UCM">
<ns1:Service IdcService="CHECKIN_UNIVERSAL">
<ns1:Document>
<ns1:Field name="dUser">WSSUSER</ns1:Field>
<ns1:Field name="dDocName">TFDLN00000005868-1400139517361</ns1:Field>
<ns1:Field name="dDocType">Document</ns1:Field>
<ns1:Field name="dSecurityGroup">Secure</ns1:Field>
<ns1:Field name="dDocAccount">USER01</ns1:Field>
<ns1:Field name="dDocTitle">TFDLN00000005868-1400139517361.txt</ns1:Field>
<ns1:Field name="dDocAuthor">WSSUSER</ns1:Field>
<ns1:Field name="dCollectionID">352885742199000201</ns1:Field>
<ns1:File name="primaryFile" href="TFDLN00000005868-1400139517361.txt">
<ns1:Contents>aGVsbG8gd29ybGQ=</ns1:Contents>
</ns1:File>
</ns1:Document>
</ns1:Service>
```

```
</GenericRequest>
```

Generic Service, Response

```
<ns2:GenericResponse xmlns:ns2="http://www.oracle.com/UCM" xmlns:wsa="http://
www.w3.org/2005/08/addressing">
<ns2:Service IdcService="CHECKIN_UNIVERSAL">
<ns2:Document>
<ns2:Field name="reserveLocation">false</ns2:Field>
<ns2:Field name="IdcService">CHECKIN_UNIVERSAL</ns2:Field>
<ns2:Field name="dSecurityGroup">Secure</ns2:Field>
<ns2:Field name="WebfilePath">/otss/dev_v110/user_projects/domains/base_domain/
ucm/cs/weblayout/groups/secure/@irreyes/documents/document/mtm5/nte3/~edisp/
tfdln00000005868-1400139517361~1.txt</ns2:Field>
<ns2:Field name="dpTriggerField">xIdcProfile</ns2:Field>
<ns2:Field name="dDocType">Document</ns2:Field>
<ns2:Field name="scriptableActionErr"/>
<ns2:Field name="xPartitionId:isSetDefault">1</ns2:Field>
<ns2:Field name="xWebFlag"/>
<ns2:Field name="dDocCreator">WSSUSER</ns2:Field>
<ns2:Field name="dDocAccount">USER01</ns2:Field>
<ns2:Field name="xClbraUserList"/>
<ns2:Field name="dLocation"/>
<ns2:Field name="xWCPageId:isSetDefault">1</ns2:Field>
<ns2:Field name="dWebExtension">txt</ns2:Field>
<ns2:Field name="isNew">1</ns2:Field>
<ns2:Field name="IsAutoNumber">1</ns2:Field>
<ns2:Field name="xReadOnly">FALSE</ns2:Field>
<ns2:Field name="xWCTags"/>
<ns2:Field name="LockedContents1">dDocName:TFDLN00000005868-1400139517361</ns2:Field>
<ns2:Field name="dClbraName"/>
<ns2:Field name="dChildManipulation">1</ns2:Field>
<ns2:Field name="dCreateDate">5/15/14 3:38 AM</ns2:Field>
<ns2:Field name="xClbraAliasList"/>
<ns2:Field name="dActionMillis">481237532</ns2:Field>
<ns2:Field name="dCollectionModifier">WSSUSER</ns2:Field>
<ns2:Field name="StatusMessage">Successfully checked in content item
 'TFDLN00000005868-1400139517361'.</ns2:Field>
<ns2:Field name="xCollectionID">35288574219900201</ns2:Field>
<ns2:Field name="isInfoOnly"/>
<ns2:Field name="dIsPrimary">1</ns2:Field>
<ns2:Field name="dActionDate">5/15/14 3:38 AM</ns2:Field>
<ns2:Field name="xWCPageId"/>
<ns2:Field name="dDocID">6842</ns2:Field>
<ns2:Field name="dPublishState"/>
<ns2:Field name="dCollectionType">0</ns2:Field>
<ns2:Field name="numlevels">0</ns2:Field>
<ns2:Field name="dRevisionID">1</ns2:Field>
<ns2:Field name="primaryFile">TFDLN00000005868-1400139517361.txt</ns2:Field>
<ns2:Field name="scriptableActionFunction">determineCheckin</ns2:Field>
<ns2:Field name="xExternalDataSet"/>
<ns2:Field name="noDocLock">1</ns2:Field>
<ns2:Field name="dID">8020</ns2:Field>
<ns2:Field name="xPartitionId"/>
<ns2:Field name="xWCTags:isSetDefault">1</ns2:Field>
<ns2:Field name="xComments:isSetDefault">1</ns2:Field>
<ns2:Field name="dInDate">5/15/14 3:38 AM</ns2:Field>
<ns2:Field name="dDocOwner">WSSUSER</ns2:Field>
<ns2:Field name="dUser">WSSUSER</ns2:Field>
<ns2:Field name="isDocProfileUsed">true</ns2:Field>
<ns2:Field name="dCollectionGUID">6FAE748F-4409-E572-E3FB-A50A9ADA434D</ns2:Field>
<ns2:Field name="scriptableActionParams"/>
<ns2:Field name="dpEvent">OnImport</ns2:Field>
<ns2:Field name="xExternalDataSet:isSetDefault">1</ns2:Field>
<ns2:Field name="dDocLastModifier">WSSUSER</ns2:Field>
<ns2:Field name="DocExists"/>
<ns2:Field name="dCollectionEnabled">1</ns2:Field>
```

```
<ns2:Field name="dConversion">PassThru</ns2:Field>
<ns2:Field name="StatusCode">0</ns2:Field>
<ns2:Field name="xStorageRule:isSetDefault">1</ns2:Field>
<ns2:Field name="dStatus">DONE</ns2:Field>
<ns2:Field name="dOriginalName">TFDLN00000005868-1400139517361.txt</ns2:Field>
<ns2:Field name="dCollectionQueries">1</ns2:Field>
<ns2:Field name="dLastModifiedDate">{ts '2013-07-22 22:30:00.000'}</ns2:Field>
<ns2:Field name="dRevClassID">7620</ns2:Field>
<ns2:Field name="dDocName">TFDLN00000005868-1400139517361</ns2:Field>
<ns2:Field name="dRevLabel">1</ns2:Field>
<ns2:Field name="idcToken"/>
<ns2:Field name="xWCWorkflowApproverUserList:isSetDefault">1</ns2:Field>
<ns2:Field name="xStorageRule">DispByContentId</ns2:Field>
<ns2:Field name="localizedForResponse">1</ns2:Field>
<ns2:Field name="dPublishType"/>
<ns2:Field name="dCollectionInherit">0</ns2:Field>
<ns2:Field name="xForceFolderSecurity">FALSE</ns2:Field>
<ns2:Field name="dParentCollectionID">35288574219900002</ns2:Field>
<ns2:Field name="dFormat">text/plain</ns2:Field>
<ns2:Field name="scriptableActionType">3</ns2:Field>
<ns2:Field name="dDocTitle">TFDLN00000005868-1400139517361.txt</ns2:Field>
<ns2:Field name="dpAction">CheckinNew</ns2:Field>
<ns2:Field name="primaryFile:path">/otss/dev_v110/user_projects/domains/base_domain/ucm/cs/
vault/~temp/1833286365.txt</ns2:Field>
<ns2:Field name="dCollectionID">35288574219900201</ns2:Field>
<ns2:Field name="xInhibitUpdate">FALSE</ns2:Field>
<ns2:Field name="refreshMonikers"/>
<ns2:Field name="xWCWorkflowAssignment"/>
<ns2:Field name="isEditMode">1</ns2:Field>
<ns2:Field name="refreshSubMonikers"/>
<ns2:Field name="xHidden">FALSE</ns2:Field>
<ns2:Field name="dCollectionOwner">WSSUSER</ns2:Field>
<ns2:Field name="changedMonikers"/>
<ns2:Field name="dRevRank">0</ns2:Field>
<ns2:Field name="dPromptForMetadata">0</ns2:Field>
<ns2:Field name="StorageRule">DispByContentId</ns2:Field>
<ns2:Field name="xComments"/>
<ns2:Field name="xWCWorkflowApproverUserList"/>
<ns2:Field name="dAction">Checkin</ns2:Field>
<ns2:Field name="isDocProfileDone">1</ns2:Field>
<ns2:Field name="xWebFlag:isSetDefault">1</ns2:Field>
<ns2:Field name="dRawDocID">6841</ns2:Field>
<ns2:Field name="xIdcProfile:isSetDefault">1</ns2:Field>
<ns2:Field name="dProcessingState">Y</ns2:Field>
<ns2:Field name="dWorkflowState"/>
<ns2:Field name="dDocCreatedDate">{ts '2014-05-15 03:38:37.459'}</ns2:Field>
<ns2:Field name="dDocAuthor">WSSUSER</ns2:Field>
<ns2:Field name="dOutDate"/>
<ns2:Field name="dIsWebFormat">0</ns2:Field>
<ns2:Field name="scriptableActionFlags">12</ns2:Field>
<ns2:Field name="isCheckin">1</ns2:Field>
<ns2:Field name="hasCollectionID">1</ns2:Field>
<ns2:Field name="RenditionId">webViewableFile</ns2:Field>
<ns2:Field name="dExtension">txt</ns2:Field>
<ns2:Field name="prevReleaseState"/>
<ns2:Field name="VaultfilePath">/otss/dev_v110/user_projects/domains/base_domain/ucm/cs/
vault/document/@irreyes/mtm5/nte3/8020.txt</ns2:Field>
<ns2:Field name="dCollectionCreator">WSSUSER</ns2:Field>
<ns2:Field name="dCollectionName">Portal Uploads</ns2:Field>
<ns2:Field name="isStatusChanged">1</ns2:Field>
<ns2:Field name="dReleaseState">N</ns2:Field>
<ns2:Field name="dWebOriginalName">TFDLN00000005868-1400139517361~1.txt</ns2:Field>
<ns2:Field name="xWCWorkflowAssignment:isSetDefault">1</ns2:Field>
<ns2:Field name="dDocLastModifiedDate">{ts '2014-05-15 03:38:37.459'}</ns2:Field>
<ns2:Field name="dFileSize">11</ns2:Field>
<ns2:Field name="xIdcProfile"/>
</ns2:Document>
</ns2:Service>
```

```
</ns2:GenericResponse>
```

UploadSupportingDocument Request (Integration Layer to Revenue Management)

This is a sample request that the integration layer forwards to the revenue management system. It contains the file location URL and the confirmation ID for the upload transaction. Optionally, it also includes the original file contents.

```
<TSUploadSupportingDocument xmlns:ns0="http://ouaf.oracle.com/" xmlns:ouaf="urn:oracle:ouaf"
 xmlns:otss="http://oracle.com/TSUploadSupportingDocument.xsd" dateTimeTagFormat=""
 xmlns="http://oracle.com/TSUploadSupportingDocument.xsd">
<otss:head>
<otss:key1>
<otss:name>PER_ID</otss:name>
<otss:value>6638893506</otss:value>
</otss:key1>
<otss:key2>
<otss:name>TAX_ROLE_ID</otss:name>
<otss:value>9037315275</otss:value>
</otss:key2>
<otss:key3/>
<otss:key4/>
<otss:key5/>
<otss:key6/>
<otss:key7/>
<otss:key8/>
<otss:key9/>
<otss:key10/>
<otss:accessType>TAXROLE</otss:accessType>
<otss:webUserId>USER01</otss:webUserId>
<otss:webUserName>IRREYES</otss:webUserName>
<otss:emailAddress/>
<otss:ipAddress>10.186.253.49</otss:ipAddress>
</otss:head>
<otss:errorMessage>
<otss:messageParameters/>
</otss:errorMessage>
<otss:confirmationData>
<otss:confirmationId>FUID56800000</otss:confirmationId>
<otss:header>
<otss:messageParameters/>
</otss:header>
</otss:confirmationData>
<otss:mainData>
<otss:documentLocator>TFDLN00000005868</otss:documentLocator>
<otss:fileURL>http://[port]:[host]/cs/Contribution%20Folders/Portal%20Uploads/
TFDLN00000005868-1400139517361.txt</otss:fileURL>
<otss:content>aGVsbG8gd29ybGQ=</otss:content>
</otss:mainData>
</TSUploadSupportingDocument>
```

UploadSupportingDocument Response (Revenue Management to Self Service)

This is a sample response from the revenue management system. It contains confirmation details.

```
<TSUploadSupportingDocument xmlns="http://oracle.com/TSUploadSupportingDocument.xsd"
 dateTimeTagFormat="">
<head>
<key1>
<name>PER_ID</name>
<value>6638893506</value>
</key1>
<key2>
```

```
<name>TAX_ROLE_ID</name>
<value>9037315275</value>
</key2>
<key3/>
<key4/>
<key5/>
<key6/>
<key7/>
<key8/>
<key9/>
<key10/>
<webUserId>USER01</webUserId>
<webUserName>John Doe</webUserName>
<emailAddress/>
<ipAddress>10.186.253.49</ipAddress>
<accessType>TAXROLE</accessType>
</head>
<errorMessage>
<messageParameters/>
</errorMessage>
<confirmationData>
<confirmationId>FUID56800000</confirmationId>
<header>
<messageParameters/>
</header>
<details>
<sequence>1</sequence>
<messageCategory>11126</messageCategory>
<messageNumber>20108</messageNumber>
<messageParameters>
<parameters>
<sequence>1</sequence>
<parameterType>STRING</parameterType>
<parameterValue>TFDLN00000005868-1400139517361.txt</parameterValue>
</parameters>
<parameters>
<sequence>2</sequence>
<parameterType>STRING</parameterType>
<parameterValue>FUID56800000</parameterValue>
</parameters>
<parameters>
<sequence>3</sequence>
<parameterType>STRING</parameterType>
<parameterValue>TFID57600000</parameterValue>
</parameters>
</messageParameters>
</details>
</confirmationData>
<mainData>
<documentLocator>TFDLN00000005868</documentLocator>
<fileURL>http://[port]:[host]/cs/Contribution%20Folders/Portal%20Uploads/
TFDLN00000005868-1400139517361.txt</fileURL>
<content>aGVsbG8gd29ybGQ=</content>
</mainData>
</TSUploadSupportingDocument>
```

Available Form Types

RetrieveActiveFormTypes Request

The request contains the user ID, form category, and the language currently used on the self service portal.

```
<ns1:TSRetrieveActiveFormTypes xmlns:ns1="http://oracle.com/TSRetrieveActiveFormTypes.xsd">
```

```
<ns1:head>
<ns1:key1/
<ns1:accessType/>
<ns1:webUserId>FORMADMUSER</ns1:webUserId>
<ns1:webUserName>Forms Administrator</ns1:webUserName>
<ns1:emailAddress/>
<ns1:ipAddress>10.154.184.11</ns1:ipAddress>
</ns1:head>
<ns1:mainData>
<ns1:formSubType>TAXFORM</ns1:formSubType>
<ns1:language>en</ns1:language>
<ns1:description/>
</ns1:mainData>
</ns1:TSRetrieveActiveFormTypes>
```

RetrieveActiveFormTypes Response

The response contains list of forms whose definitions are available for import.

```
<TSRetrieveActiveFormTypes xmlns:schemans2="http://oracle.com/TSRetrieveActiveFormTypes.xsd"
 dateTimeTagFormat="" xmlns="http://oracle.com/TSRetrieveActiveFormTypes.xsd">
<schemans2:head>
<schemans2:key1/>
<schemans2:key2/>
<schemans2:key3/>
<schemans2:key4/>
<schemans2:key5/>
<schemans2:key6/>
<schemans2:key7/>
<schemans2:key8/>
<schemans2:key9/>
<schemans2:key10/>
<schemans2:accessType/>
<schemans2:webUserId>FORMADMUSER</schemans2:webUserId>
<schemans2:webUserName>Forms Administrator</schemans2:webUserName>
<schemans2:emailAddress/>
<schemans2:ipAddress>10.154.184.11</schemans2:ipAddress>
</schemans2:head>
<schemans2:requestStatus/>
<schemans2:confirmationData>
<schemans2:confirmationId/>
</schemans2:confirmationData>
<schemans2:mainData>
<schemans2:formSubType>TAXFORM </schemans2:formSubType>
<schemans2:language>en</schemans2:language>
<schemans2:startDate/>
<schemans2:endDate/>
<schemans2:description>%</schemans2:description>
<schemans2:forms>
<schemans2:formType>CORPINC2008</schemans2:formType>
<schemans2:startDate>2008-01-01</schemans2:startDate>
<schemans2:endDate>2008-12-31</schemans2:endDate>
<schemans2:description>2008 Corporate Income Tax Form </schemans2:description>
</schemans2:forms>
<schemans2:forms>
<schemans2:formType>INDIV2008</schemans2:formType>
<schemans2:startDate>2008-01-01</schemans2:startDate>
<schemans2:endDate>2008-12-31</schemans2:endDate>
<schemans2:description>2008 Individual Income Tax Return</schemans2:description>
</schemans2:forms>
</schemans2:mainData>
</TSRetrieveActiveFormTypes>
```

Import Form Definitions

RetrieveFormTypeDefinitions Request

This example illustrates a scenario in which a single form is selected for import.

```
<ns1:TSRetrieveFormTypeDefinitions xmlns:ns1="http://oracle.com/
TSRetrieveFormTypeDefinitions.xsd">
<ns1:head>
<ns1:key1/>
<ns1:key2/>
<ns1:accessType/>
<ns1:webUserId>FORMADMUSER</ns1:webUserId>
<ns1:webUserName>Forms Administrator</ns1:webUserName>
<ns1:emailAddress/>
<ns1:ipAddress>10.154.98.232</ns1:ipAddress>
</ns1:head>
<ns1:mainData>
<ns1:input>
<ns1:formType>CORPINC2010</ns1:formType>
<ns1:language>en</ns1:language>
</ns1:input>
</ns1:mainData>
</ns1:TSRetrieveFormTypeDefinitions>
```

RetrieveFormTypeDefinitions Response

This example shows an import of a single form with two sections.

```
<TSRetrieveFormTypeDefinitions xmlns:schemans2="http://oracle.com/
 TSRetrieveFormTypeDefinitions.xsd" dateTimeTagFormat="">
<schemans2:head>
<schemans2:key1/>
<schemans2:key2/>
<schemans2:key3/>
<schemans2:key4/>
<schemans2:key5/>
<schemans2:key6/>
<schemans2:key7/>
<schemans2:key8/>
<schemans2:key9/>
<schemans2:key10/>
<schemans2:accessType>TAXROLE</schemans2:accessType>
<schemans2:webUserId>FORMADMUSER</schemans2:webUserId>
<schemans2:webUserName>Forms Administrator</schemans2:webUserName>
<schemans2:emailAddress/>
<schemans2:ipAddress>10.154.98.232</schemans2:ipAddress>
</schemans2:head>
<schemans2:requestStatus xmlns:schemans2="http://oracle.com/
TSRetrieveFormTypeDefinitions.xsd"/>
<schemans2:confirmationData xmlns:schemans2="http://oracle.com/
TSRetrieveFormTypeDefinitions.xsd">
<schemans2:confirmationId/>
</schemans2:confirmationData>
<schemans2:mainData xmlns:schemans2="http://oracle.com/TSRetrieveFormTypeDefinitions.xsd">
<schemans2:input>
<schemans2:formType>CORPINC2010</schemans2:formType>
<schemans2:language>en</schemans2:language>
</schemans2:input>
<schemans2:output>
<schemans2:formDef>
```

```
<schemans2:formType>CORPINC2010</schemans2:formType>
<schemans2:startDate>2010-01-01</schemans2:startDate>
<schemans2:endDate>2010-12-31</schemans2:endDate>
<schemans2:description>
<schemans2:languages>
<schemans2:text>2010 Corporate Income Tax Form</schemans2:text>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:description>
<schemans2:section>
<schemans2:name>taxpayerInfo</schemans2:name>
<schemans2:occurrence>SINGLE</schemans2:occurrence>
<schemans2:displaySequence>10</schemans2:displaySequence>
<schemans2:description>
<schemans2:languages>
<schemans2:text>Taxpayer Information</schemans2:text>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:description>
<schemans2:help>
<schemans2:languages>
<schemans2:text/>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:help>
<schemans2:line>
<schemans2:displaySequence>10</schemans2:displaySequence>
<schemans2:field>
<schemans2:name>taxYrBeginDt</schemans2:name>
<schemans2:lineNumber/>
<schemans2:dataType>DATE</schemans2:dataType>
<schemans2:fieldLength>10</schemans2:fieldLength>
<schemans2:description>
<schemans2:languages>
<schemans2:text>Tax Year Begin Date</schemans2:text>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:description>
<schemans2:help>
<schemans2:languages>
<schemans2:text/>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:help>
</schemans2:field>
</schemans2:line>
<schemans2:line>
<schemans2:displaySequence>20</schemans2:displaySequence>
<schemans2:field>
<schemans2:name>taxYrEndDt</schemans2:name>
<schemans2:lineNumber/>
<schemans2:dataType>DATE</schemans2:dataType>
<schemans2:fieldLength>10</schemans2:fieldLength>
<schemans2:description>
<schemans2:languages>
<schemans2:text>Tax Year End Date</schemans2:text>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:description>
<schemans2:help>
<schemans2:languages>
<schemans2:text/>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:help>
</schemans2:field>
</schemans2:line>
<schemans2:line>
```

```
<schemans2:displaySequence>30</schemans2:displaySequence>
<schemans2:field>
<schemans2:name>name</schemans2:name>
<schemans2:lineNumber/>
<schemans2:dataType>TEXT</schemans2:dataType>
<schemans2:fieldLength>254</schemans2:fieldLength>
<schemans2:description>
<schemans2:languages>
<schemans2:text>Name</schemans2:text>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:description>
<schemans2:help>
<schemans2:languages>
<schemans2:text/>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:help>
</schemans2:field>
</schemans2:line>
<schemans2:line>
<schemans2:displaySequence>40</schemans2:displaySequence>
<schemans2:field>
<schemans2:name>country</schemans2:name>
<schemans2:lineNumber/>
<schemans2:dataType>TEXT</schemans2:dataType>
<schemans2:fieldLength>3</schemans2:fieldLength>
<schemans2:description>
<schemans2:languages>
<schemans2:text>Country</schemans2:text>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:description>
<schemans2:help>
<schemans2:languages>
<schemans2:text/>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:help>
</schemans2:field>
</schemans2:line>
<schemans2:line>
<schemans2:displaySequence>50</schemans2:displaySequence>
<schemans2:field>
<schemans2:name>address</schemans2:name>
<schemans2:lineNumber/>
<schemans2:dataType>TEXT</schemans2:dataType>
<schemans2:fieldLength>254</schemans2:fieldLength>
<schemans2:description>
<schemans2:languages>
<schemans2:text>Address</schemans2:text>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:description>
<schemans2:help>
<schemans2:languages>
<schemans2:text/>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:help>
</schemans2:field>
</schemans2:line>
<schemans2:line>
<schemans2:displaySequence>60</schemans2:displaySequence>
<schemans2:field>
<schemans2:name>address2</schemans2:name>
<schemans2:lineNumber/>
<schemans2:dataType>TEXT</schemans2:dataType>
```

```
<schemans2:fieldLength>254</schemans2:fieldLength>
<schemans2:description>
<schemans2:languages>
<schemans2:text>Address 2</schemans2:text>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:description>
<schemans2:help>
<schemans2:languages>
<schemans2:text/>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:help>
</schemans2:field>
</schemans2:line>
<schemans2:line>
<schemans2:displaySequence>70</schemans2:displaySequence>
<schemans2:field>
<schemans2:name>city</schemans2:name>
<schemans2:lineNumber/>
<schemans2:dataType>TEXT</schemans2:dataType>
<schemans2:fieldLength>30</schemans2:fieldLength>
<schemans2:description>
<schemans2:languages>
<schemans2:text>City</schemans2:text>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:description>
<schemans2:help>
<schemans2:languages>
<schemans2:text/>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:help>
</schemans2:field>
</schemans2:line>
<schemans2:line>
<schemans2:displaySequence>80</schemans2:displaySequence>
<schemans2:field>
<schemans2:name>state</schemans2:name>
<schemans2:lineNumber/>
<schemans2:dataType>TEXT</schemans2:dataType>
<schemans2:fieldLength>6</schemans2:fieldLength>
<schemans2:description>
<schemans2:languages>
<schemans2:text>State</schemans2:text>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:description>
<schemans2:help>
<schemans2:languages>
<schemans2:text/>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:help>
</schemans2:field>
</schemans2:line>
<schemans2:line>
<schemans2:displaySequence>90</schemans2:displaySequence>
<schemans2:field>
<schemans2:name>zipCode</schemans2:name>
<schemans2:lineNumber/>
<schemans2:dataType>TEXT</schemans2:dataType>
<schemans2:fieldLength>12</schemans2:fieldLength>
<schemans2:description>
<schemans2:languages>
<schemans2:text>Postal</schemans2:text>
<schemans2:language>en</schemans2:language>
```

```
</schemans2:languages>
</schemans2:description>
<schemans2:help>
<schemans2:languages>
<schemans2:text/>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:help>
</schemans2:field>
</schemans2:line>
</schemans2:section>
<schemans2:section>
<schemans2:name>rtrnCorpInfo</schemans2:name>
<schemans2:occurrence>SINGLE</schemans2:occurrence>
<schemans2:displaySequence>20</schemans2:displaySequence>
<schemans2:description>
<schemans2:languages>
<schemans2:text>Return / Corporation Information</schemans2:text>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:description>
<schemans2:help>
<schemans2:languages>
<schemans2:text/>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:help>
<schemans2:line>
<schemans2:displaySequence>20</schemans2:displaySequence>
<schemans2:field>
<schemans2:name>consReturn</schemans2:name>
<schemans2:lineNumber>1 a</schemans2:lineNumber>
<schemans2:dataType>BOOLEAN</schemans2:dataType>
<schemans2:fieldLength>1</schemans2:fieldLength>
<schemans2:description>
<schemans2:languages>
<schemans2:text>Consolidated Return</schemans2:text>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:description>
<schemans2:help>
<schemans2:languages>
<schemans2:text/>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:help>
</schemans2:field>
</schemans2:line>
<schemans2:line>
<schemans2:displaySequence>30</schemans2:displaySequence>
<schemans2:field>
<schemans2:name>lifeNonLifeConsReturn</schemans2:name>
<schemans2:lineNumber>1 b</schemans2:lineNumber>
<schemans2:dataType>BOOLEAN</schemans2:dataType>
<schemans2:fieldLength>1</schemans2:fieldLength>
<schemans2:description>
<schemans2:languages>
<schemans2:text>Life / Non-Life Consolidated Return</schemans2:text>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:description>
<schemans2:help>
<schemans2:languages>
<schemans2:text/>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:help>
</schemans2:field>
```

```
</schemans2:line>
<schemans2:line>
<schemans2:displaySequence>40</schemans2:displaySequence>
<schemans2:field>
<schemans2:name>persHoldngCo</schemans2:name>
<schemans2:lineNumber>2</schemans2:lineNumber>
<schemans2:dataType>BOOLEAN</schemans2:dataType>
<schemans2:fieldLength>1</schemans2:fieldLength>
<schemans2:description>
<schemans2:languages>
<schemans2:text>Personal Holding Company</schemans2:text>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:description>
<schemans2:help>
<schemans2:languages>
<schemans2:text/>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:help>
</schemans2:field>
</schemans2:line>
<schemans2:line>
<schemans2:displaySequence>50</schemans2:displaySequence>
<schemans2:field>
<schemans2:name>persnlSvcCorp</schemans2:name>
<schemans2:lineNumber>3</schemans2:lineNumber>
<schemans2:dataType>BOOLEAN</schemans2:dataType>
<schemans2:fieldLength>1</schemans2:fieldLength>
<schemans2:description>
<schemans2:languages>
<schemans2:text>Personal Service Corporation</schemans2:text>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:description>
<schemans2:help>
<schemans2:languages>
<schemans2:text/>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:help>
</schemans2:field>
</schemans2:line>
<schemans2:line>
<schemans2:displaySequence>60</schemans2:displaySequence>
<schemans2:field>
<schemans2:name>netIncomeLossRecon</schemans2:name>
<schemans2:lineNumber>4</schemans2:lineNumber>
<schemans2:dataType>BOOLEAN</schemans2:dataType>
<schemans2:fieldLength>1</schemans2:fieldLength>
<schemans2:description>
<schemans2:languages>
<schemans2:text>Net Income / Loss Reconciliation Form Attached</schemans2:text>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:description>
<schemans2:help>
<schemans2:languages>
<schemans2:text/>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:help>
</schemans2:field>
</schemans2:line>
<schemans2:line>
<schemans2:displaySequence>70</schemans2:displaySequence>
<schemans2:field>
<schemans2:name>idNumber</schemans2:name>
```

```
<schemans2:lineNumber>B</schemans2:lineNumber>
<schemans2:dataType>TEXT</schemans2:dataType>
<schemans2:fieldLength>16</schemans2:fieldLength>
<schemans2:description>
<schemans2:languages>
<schemans2:text>Employer Identification Number</schemans2:text>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:description>
<schemans2:help>
<schemans2:languages>
<schemans2:text/>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:help>
</schemans2:field>
</schemans2:line>
<schemans2:line>
<schemans2:displaySequence>80</schemans2:displaySequence>
<schemans2:field>
<schemans2:name>dateIncorp</schemans2:name>
<schemans2:lineNumber>C</schemans2:lineNumber>
<schemans2:dataType>DATE</schemans2:dataType>
<schemans2:fieldLength>10</schemans2:fieldLength>
<schemans2:description>
<schemans2:languages>
<schemans2:text>Date Incorporated</schemans2:text>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:description>
<schemans2:help>
<schemans2:languages>
<schemans2:text/>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:help>
</schemans2:field>
</schemans2:line>
<schemans2:line>
<schemans2:displaySequence>120</schemans2:displaySequence>
<schemans2:field>
<schemans2:name>finalRetrn</schemans2:name>
<schemans2:lineNumber>2</schemans2:lineNumber>
<schemans2:dataType>BOOLEAN</schemans2:dataType>
<schemans2:fieldLength>1</schemans2:fieldLength>
<schemans2:description>
<schemans2:languages>
<schemans2:text>Final Return</schemans2:text>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:description>
<schemans2:help>
<schemans2:languages>
<schemans2:text/>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:help>
</schemans2:field>
</schemans2:line>
<schemans2:line>
<schemans2:displaySequence>130</schemans2:displaySequence>
<schemans2:field>
<schemans2:name>nameChng</schemans2:name>
<schemans2:lineNumber>3</schemans2:lineNumber>
<schemans2:dataType>BOOLEAN</schemans2:dataType>
<schemans2:fieldLength>1</schemans2:fieldLength>
<schemans2:description>
<schemans2:languages>
```

```
<schemans2:text>Name Change</schemans2:text>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:description>
<schemans2:help>
<schemans2:languages>
<schemans2:text/>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:help>
</schemans2:field>
</schemans2:line>
<schemans2:line>                    <schemans2:displaySequence>140</
schemans2:displaySequence>
<schemans2:field>
<schemans2:name>addrChng</schemans2:name>
<schemans2:lineNumber>4</schemans2:lineNumber>
<schemans2:dataType>BOOLEAN</schemans2:dataType>
<schemans2:fieldLength>1</schemans2:fieldLength>
<schemans2:description>
<schemans2:languages>
<schemans2:text>Address Change</schemans2:text>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:description>
<schemans2:help>
<schemans2:languages>
<schemans2:text/>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:help>
</schemans2:field>
</schemans2:line>
</schemans2:section>
</schemans2:formDef>
<schemans2:availableLanguages>
<schemans2:language>en</schemans2:language>
</schemans2:availableLanguages>
</schemans2:output>
</schemans2:mainData>
</TSRetrieveFormTypeDefinitions>
```

Refresh Lookup

The samples below illustrate a refresh of a single lookup SALES_AREA_NAME.

RefreshLookup Request

```
<ns1:TSRefreshFormLookup xmlns:ns1="http://oracle.com/TSRefreshFormLookup.xsd">
<ns1:head>
<ns1:key1/>
<ns1:key2/>
<ns1:webUserId>FORMADMUSER</ns1:webUserId>
<ns1:webUserName>Form Administrator</ns1:webUserName>
<ns1:emailAddress/>
<ns1:ipAddress>10.154.120.204</ns1:ipAddress>
</ns1:head>
<ns1:mainData>
<ns1:input>
<ns1:lookupName>SALES_AREA_NAME </ns1:lookupName>
</ns1:input>
</ns1:mainData>
</ns1:TSRefreshFormLookup>
```

RefreshLookup Response

```
<TSRefreshFormLookup xmlns:schemans2="http://oracle.com/TSRefreshFormLookup.xsd"
 dateTimeTagFormat="" xmlns="http://oracle.com/TSRefreshFormLookup.xsd">
<schemans2:head>
<schemans2:key1/>
<schemans2:key2/>
<schemans2:key3/>
<schemans2:key4/>
<schemans2:key5/>
<schemans2:key6/>
<schemans2:key7/>
<schemans2:key8/>
<schemans2:key9/>
<schemans2:key10/>
<schemans2:webUserId>FORMADMUSER</schemans2:webUserId>
<schemans2:webUserName>Form Administrator</schemans2:webUserName>
<schemans2:emailAddress/>
<schemans2:ipAddress>10.154.113.115</schemans2:ipAddress>
</schemans2:head>
<schemans2:requestStatus/>
<schemans2:confirmationData>
<schemans2:confirmationId/>
</schemans2:confirmationData>
<schemans2:mainData>
<schemans2:input>
<schemans2:lookupName> SALES_AREA_NAME</schemans2:lookupName>
</schemans2:input>
<schemans2:output>
<schemans2:values>
<schemans2:value>CMA1</schemans2:value>
<schemans2:description>
<schemans2:languages>
<schemans2:text>Area 1</schemans2:text>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:description>
</schemans2:values>
<schemans2:values>
<schemans2:value>CMA2</schemans2:value>
<schemans2:description>
<schemans2:languages>
<schemans2:text>Area 2</schemans2:text>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:description>
</schemans2:values>
<schemans2:values>
<schemans2:value>CMA3</schemans2:value>
<schemans2:description>
<schemans2:languages>
<schemans2:text>Area 3</schemans2:text>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:description>
</schemans2:values>
<schemans2:values>
<schemans2:value>CMA4</schemans2:value>
<schemans2:description>
<schemans2:languages>
<schemans2:text>Area 4</schemans2:text>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:description>
</schemans2:values>
<schemans2:values>
```

```
<schemans2:value>CMA5</schemans2:value>
<schemans2:description>
<schemans2:languages>
<schemans2:text>Area 5</schemans2:text>
<schemans2:language>en</schemans2:language>
</schemans2:languages>
</schemans2:description>
</schemans2:values>
</schemans2:output>
</schemans2:mainData>
</TSRefreshFormLookup>
```

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: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** 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

Official Payments Post-back XML Message			TSOneTimePayment Message			DVM Mapping
						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	
suffix	name	Field	suffix	name	Field	
customDataElement	customDataElements	Field	businessName	externalPaymentData	Field	Populate with customDataElement with attribute sequenceNumber = 4
address	PaymentPostBack	Group	address	externalPaymentData	Group	
street1	address	Field	street1	address	Field	
street2	address	Field	street2	address	Field	
city	address	Field	city	address	Field	

Official Payments Post-back XML Message			TSOneTimePayment Message			DVM Mapping
stateOrProvince	address	Field	state	address	Field	
zipOrPostalCode	address	Field	postal	address	Field	
countryCode	address	Field	country	address	Field	**OTSS_Country**
emailAddress	PaymentPostBack	Field	emailAddress	externalPaymentData	Field	
phoneNumber	PaymentPostBack	Field	phoneNumber	externalPaymentData	Field	
receiptNumber	PaymentPostBack	Field	recieptNumber	externalPaymentData	Field	Same as an authorization code
paymentID	PaymentPostBack	Field	paymentID	externalPaymentData	Field	
trafficSchool	PaymentPostBack	Field	trafficSchool	externalPaymentData	Field	
paymentChannel	PaymentPostBack	Field	paymentChannel	externalPaymentData	Field	

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	GroupRecord	
					paymentVendor	paymentReportRecord	Field	This value be picked from Configuration Properties
					currency	paymentReportRecord	Field	Default the Currency code from Configuration Properties
1	Custom Data Element #1	75	Alphanumeric		payDestinationType	paymentReportRecord	Field	**OTSS_PaymentDestination** Populate with customDataElement with attribute sequenceNumber = 1
2	Custom Data Element #2	75	Alphanumeric					
3	Custom Data Element #3	75	Alphanumeric					
4	Custom Data Element #4	75	Alphanumeric		businessName	externalPaymentReportData	Field	Populate with customDataElement

							Mapping
							with attribute sequenceNumber = 4
5	Custom Data Element #5	75	Alphanumeric		externalID	paymentReportRecord	Populate with customDataElement with attribute sequenceNumber = 5
					destinationDetails	paymentReportRecord	Populated in a loop
					sequence	destinationDetails	Populate the sequenceNumber from 1 to 4 in a loop
					fieldName	destinationDetails	
6	Custom Data Element #6	75	Alphanumeric		fieldValue	destinationDetails	Populate the values for customDataElement in a loop where sequenceNumber = 6
7	Custom Data Element #7	75	Alphanumeric		fieldValue	destinationDetails	Populate the values for customDataElement in a loop where sequenceNumber =7
8	Custom Data Element #8	75	Alphanumeric		fieldValue	destinationDetails	Populate the values for customDataElement in a loop where sequenceNumber =8
9	Custom Data Element #9	75	Alphanumeric		fieldValue	destinationDetails	Populate the values for customDataElement in a loop where sequenceNumber = 9
10	Date of Transaction	8	Numeric	Format: yyyymmdd; Example: 20030531	paymentDate	paymentReportRecord	Convert to xsd:Date

Official Payments Payment Report File Format					TSProcessExtPayReportRecord Message		DVM Mapping
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				
					externalPaymentReportData	paymentReportRecord	
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

								Mapping
								from OPC mapped to two different fields OPC_ PaymentType and maps to accountType from OPC
					paymentType	paymentReportRecord	Field	**OPC_ PaymentType**
24	Authorization Code	50	Alphanumeric	Examples: 123456, T34997	extTransactionRefID	paymentReportRecord	Field	
25	Receipt Number	50	Alphanumeric	Generally the same as the Authorization Code unless Receipt Number is otherwise provided or specified.	recieptNumber	externalPaymentReportData	Field	
26	Traffic School Flag	4	Numeric	(0=No TS; 1..9=Fee in Payment Amt. #x)	trafficSchool	externalPaymentReportData	Field	
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	Field	
29	ANI	20	Numeric	IVR only - caller's phone number as collected via caller-ID				
					name	externalPaymentReportData	Field	
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	

Official Payments Payment Report File Format					TSProcessExtPayReportRecord Message			DVM Mapping
					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	
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	Field	
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	Field	
44	Unique Identifier	50	Alphanumeric	Used for Co-Brand + and STP only. Unique Identifier for the transaction as provided by the Client.	uniqueIdentifier	externalPaymentReportData	Field	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.
Base product, "out-of-the-box" product	PSRMSS base product as delivered to the revenue management authority.
BPEL	Business Process Execution Language.
Casual user	Taxpayer who may use a limited set of features that do not require registration or login.
Confirmation number	An automatically-generated unique reference identifier provided to the taxpayer on all transactions and service requests.
DDL	Data Definition Language.
DVM	Domain value maps operate on actual data values that transit through the infrastructure at runtime.
EM/OEM	Oracle Enterprise Manager
Enrolled user	Registered user who requested and was granted the ability to manage all or some of the information kept in the revenue management system through the self-service portal.
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.
Implementer, implementer	Those who implement PSRMSS for the revenue management authority.
ITA	Interactive Tax Assistant
JSF	Java Server Faces

Term/Abbreviation	Definition
Line of Business (LOB)	Broad category of self-service site users with similar interests; used throughout the portal to group the services and features appropriately. Examples: Individual, Business.
Managed Content	Documents that are available through WebCenter Content.
MDS	Metadata Services
Oracle BPM Worklist	Enables business users to access and act on tasks assigned to them.
ODS	Oracle Determinations Server. Part of Oracle Policy Automation.
OIM	Oracle Identity Management
OPA	Oracle Policy Automation. Automates complex policies, delivering self-service guidance and high volume decision making in both batch and real-time environments.
OPC	Official Payments Corporation. (See "External payment provider" definition)
OUC	Oracle Unified Content
OWD	Oracle Web Determinations. Part of Oracle Policy Automation.
OUD	Oracle Unified Directory
OWSM	Oracle Web Services Manager
POI	Proof of identity. Identification details required only for casual (not signed in) website users. (Also see "taxpayer identification".)
PSRM	Oracle Public Sector Revenue Management
PSRMSS	Oracle Public Sector Revenue Management Self Service
Registered user	Web portal visitor who established a user ID and password via the Identity Manager. This user is authenticated upon login and may access secured pages that are not accessible for a casual user.
Revenue management authority	Public sector entities that manage taxation and/or revenue.
Revenue management system (or "back-end" system)	Software application(s) used for revenue management. The Web Self-Service Portal provides public access to a limited set of functions in this system.
Session context	Basic information (user name, selected taxpayer, account, etc.) available to all portal pages throughout a single session.
SOA	Service Oriented Architecture. (Also see SOA Composer.)
SOA Composer (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.
UCM Documents	Documents on the UCM server.
WC	Oracle WebCenter
WebCenter Administration Console	A part of the WebCenter application framework that 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.
WLS	Oracle WebLogic Server
WSDL	Web Services Description Language. XML-based interface description language that is used for describing the functionality offered by a web service.

Appendix F

Print Form Data - Print Custom Transformation

This example illustrates how a generic form data sample is transformed into Form-specific XML using XSLT

Sample Form Data

This XML fragment contains generic form data and language-specific labels. It represents the original input for the form print service.

```
<documentData>
<formType>SALESDST2008</formType>
<formCategory>TAXFORM</formCategory>
<currency/>
<formTypeDescription>Sales and Use (w/ Local Distribution) Tax Form</formTypeDescription>
<documentLocator/>
<documentLocatorLabel>Document Locator</documentLocatorLabel>
<customProperties>
<sequence>1</sequence>
<propertyName>CITY</propertyName>
<propertyDescription>City</propertyDescription>
<value>Tacoma</value>
</customProperties>
<customProperties>
<sequence>2</sequence>
<propertyName>CITY</propertyName>
<propertyDescription>City</propertyDescription>
<value>London</value>
</customProperties>
<formData>
<id>SALESDST2008</id>
<name>SALESDST2008</name>
<section>
<id>75560</id>
<name>taxpayerInfo</name>
```

```
<sectionLabel>Taxpayer Information</sectionLabel>
<sequence>1</sequence>
<line>
<field>
<id>75572</id>
<name>filingEndDate</name>
<lineLabel>Filing Period End Date</lineLabel>
<dataType>DATE</dataType>
<value>03/31/2012</value>
</field>
</line>
<line>
<field>
<id>75561</id>
<name>name</name>
<lineLabel>Name</lineLabel>
<dataType>TEXT</dataType>
<value>Cooks For a Cause</value>
</field>
</line>
<line>
<field>
<id>75562</id>
<name>idType</name>
<lineLabel>ID Type</lineLabel>
<dataType>TEXT</dataType>
<value>EIN</value>
</field>
</line>
<line>
<field>
<id>75563</id>
<name>idNumber</name>
<lineLabel>ID Number</lineLabel>
<dataType>TEXT</dataType>
<value>55-8947563</value>
</field>
</line>
<line>
<field>
<id>75564</id>
<name>country</name>
<lineLabel>Country</lineLabel>
<dataType>TEXT</dataType>
<value>USA</value>
</field>
</line>
<line>
<field>
<id>75565</id>
<name>address</name>
<lineLabel>Address</lineLabel>
<dataType>TEXT</dataType>
<value>1 main st.</value>
</field>
</line>
<line>
<field>
<id>75566</id>
<name>address2</name>
<lineLabel>Address 2</lineLabel>
<dataType>TEXT</dataType>
<value/>
</field>
</line>
<line>
<field>
<id>75567</id>
```

```
<name>city</name>
<lineLabel>City</lineLabel>
<dataType>TEXT</dataType>
<value>Midwood</value>
</field>
</line>
<line>
<field>
<id>75568</id>
<name>state</name>
<lineLabel>State</lineLabel>
<dataType>TEXT</dataType>
<value>VA</value>
</field>
</line>
<line>
<field>
<id>75569</id>
<name>zipCode</name>
<lineLabel>Postal</lineLabel>
<dataType>TEXT</dataType>
<value>09809</value>
</field>
</line>
<line>
<field>
<id>75570</id>
<name>accountNumber</name>
<lineLabel>Account</lineLabel>
<dataType>TEXT</dataType>
<value/>
</field>
</line>
<line>
<field>
<id>75571</id>
<name>filingStartDate</name>
<lineLabel>Filing Period Start Date</lineLabel>
<dataType>DATE</dataType>
<value>01/01/2012</value>
</field>
</line>
</section>
<section>
<id>75573</id>
<name>lineItems</name>
<sectionLabel>Line Items</sectionLabel>
<sequence>1</sequence>
<line>
<field>
<id>75574</id>
<name>totalGrossSales</name>
<lineLabel>Total Gross Sales</lineLabel>
<dataType>CURRENCY</dataType>
<value>120000.0</value>
</field>
</line>
<line>
<field>
<id>75581</id>
<name>transactionsSubjectToLocalTax</name>
<lineLabel>Transactions Subject To Local Tax</lineLabel>
<dataType>CURRENCY</dataType>
<value>0.0</value>
</field>
</line>
<line>
<field>
```

```
<id>75575</id>
<name>purchasesSubjectToTax</name>
<lineLabel>Purchases Subject to Tax</lineLabel>
<dataType>CURRENCY</dataType>
<value>2543.0</value>
</field>
</line>
<line>
<field>
<id>75576</id>
<name>total</name>
<lineLabel>Total</lineLabel>
<dataType>CURRENCY</dataType>
<value>122543.0</value>
</field>
</line>
<line>
<field>
<id>75577</id>
<name>totalExemptTransactions</name>
<lineLabel>Total Deductions / Exemptions</lineLabel>
<dataType>CURRENCY</dataType>
<value>0.0</value>
</field>
</line>
<line>
<field>
<id>75578</id>
<name>taxableTransactions</name>
<lineLabel>Transactions Subject To Tax</lineLabel>
<dataType>CURRENCY</dataType>
<value>122543.0</value>
</field>
</line>
<line>
<field>
<id>75579</id>
<name>taxRate</name>
<lineLabel>Tax Rate</lineLabel>
<dataType>NUMBER</dataType>
<value>10.0</value>
</field>
</line>
<line>
<field>
<id>75580</id>
<name>totalAssessedTaxAmount</name>
<lineLabel>Tax</lineLabel>
<dataType>CURRENCY</dataType>
<value>12254.3</value>
</field>
</line>
<line>
<field>
<id>75582</id>
<name>localTax</name>
<lineLabel>Local Tax</lineLabel>
<dataType>CURRENCY</dataType>
<value>0.0</value>
</field>
</line>
<line>
<field>
<id>75583</id>
<name>totalTax</name>
<lineLabel>Total Tax</lineLabel>
<dataType>CURRENCY</dataType>
<value>12254.3</value>
```

```
</field>
</line>
<line>
<field>
<id>75584</id>
<name>firstPrepayment</name>
<lineLabel>First Prepayment</lineLabel>
<dataType>CURRENCY</dataType>
<value>0.0</value>
</field>
</line>
<line>
<field>
<id>75585</id>
<name>secondPrepayment</name>
<lineLabel>Second Prepayment</lineLabel>
<dataType>CURRENCY</dataType>
<value>0.0</value>
</field>
</line>
<line>
<field>
<id>75586</id>
<name>taxPrepayments</name>
<lineLabel>Total Tax Prepayments</lineLabel>
<dataType>CURRENCY</dataType>
<value>0.0</value>
</field>
</line>
<line>
<field>
<id>75587</id>
<name>remainingTaxDue</name>
<lineLabel>Remaining Tax Due</lineLabel>
<dataType>CURRENCY</dataType>
<value>12254.3</value>
</field>
</line>
<line>
<field>
<id>75588</id>
<name>penalty</name>
<lineLabel>Penalty</lineLabel>
<dataType>CURRENCY</dataType>
<value>0.0</value>
</field>
</line>
<line>
<field>
<id>75589</id>
<name>interest</name>
<lineLabel>Interest</lineLabel>
<dataType>CURRENCY</dataType>
<value>0.0</value>
</field>
</line>
<line>
<field>
<id>75590</id>
<name>totalAmountOwed</name>
<lineLabel>Total Amount Owed</lineLabel>
<dataType>CURRENCY</dataType>
<value>12254.3</value>
</field>
</line>
</section>
<section>
<id>75597</id>
```

```
<name>deductExempt</name>
<sectionLabel>Schedule A - Deductions / Exemptions</sectionLabel>
<sequence>1</sequence>
<line>
<field>
<id>75598</id>
<name>salesToOtherRetailers</name>
<lineLabel>Sales to Other Retailers</lineLabel>
<dataType>CURRENCY</dataType>
<value>0.0</value>
</field>
</line>
<line>
<field>
<id>75599</id>
<name>nonTaxableSales</name>
<lineLabel>Non-Taxable Sales</lineLabel>
<dataType>CURRENCY</dataType>
<value>0.0</value>
</field>
</line>
<line>
<field>
<id>75600</id>
<name>nonTaxableLabor</name>
<lineLabel>Non-Taxable Labor</lineLabel>
<dataType>CURRENCY</dataType>
<value>0.0</value>
</field>
</line>
<line>
<field>
<id>75601</id>
<name>salesToGovernment</name>
<lineLabel>Sales to Government</lineLabel>
<dataType>CURRENCY</dataType>
<value>0.0</value>
</field>
</line>
<line>
<field>
<id>75602</id>
<name>salesInterstateForeignCommer</name>
<lineLabel>Sales in Interstate Foreign Commerce</lineLabel>
<dataType>CURRENCY</dataType>
<value>0.0</value>
</field>
</line>
<line>
<field>
<id>75603</id>
<name>salesTaxInGrossSales</name>
<lineLabel>Sales Tax in Gross Sales</lineLabel>
<dataType>CURRENCY</dataType>
<value>0.0</value>
</field>
</line>
<line>
<field>
<id>75604</id>
<name>lossesTaxableSales</name>
<lineLabel>Bad Debt Losses on Taxable Sales</lineLabel>
<dataType>CURRENCY</dataType>
<value>0.0</value>
</field>
</line>
<line>
<field>
```

```
<id>75605</id>
<name>lenderLosses</name>
<lineLabel>Bad Debt Lender Losses</lineLabel>
<dataType>CURRENCY</dataType>
<value>0.0</value>
</field>
</line>
<line>
<field>
<id>75606</id>
<name>taxPaidPurchaseResold</name>
<lineLabel>Cost of Tax-Paid Purchases Resold Prior to Use</lineLabel>
<dataType>CURRENCY</dataType>
<value>0.0</value>
</field>
</line>
<line>
<field>
<id>75607</id>
<name>returnedTaxableMerchandise</name>
<lineLabel>Returned Taxable Merchandise</lineLabel>
<dataType>CURRENCY</dataType>
<value>0.0</value>
</field>
</line>
<line>
<field>
<id>75608</id>
<name>cashDiscountsTaxableSales</name>
<lineLabel>Cash Discounts on Taxable Sales</lineLabel>
<dataType>CURRENCY</dataType>
<value>0.0</value>
</field>
</line>
<line>
<field>
<id>75609</id>
<name>totalFullDeductExempt</name>
<lineLabel>Total Deductions / Exemptions</lineLabel>
<dataType>CURRENCY</dataType>
<value>0.0</value>
</field>
</line>
</section>
<section>
<id>75610</id>
<name>localTaxCompSchedule</name>
<sectionLabel>Schedule B - Local Tax Computation</sectionLabel>
<sequence>1</sequence>
<table>
<id>75615</id>
<name>compArea</name>
<tableLabel>Computation for Local Area</tableLabel>
<tableRow>
<sequence>1</sequence>
<field>
<id>75616</id>
<name>localAreaName</name>
<fieldLabel>Local Area</fieldLabel>
<dataType>LOOKUP</dataType>
<value>Denvile</value>
</field>
<field>
<id>75618</id>
<name>localAreaTaxRate</name>
<fieldLabel>Local Area Tax Rate</fieldLabel>
<dataType>NUMBER</dataType>
<value>0.05</value>
```

```
</field>
<field>
<id>75619</id>
<name>areaTaxDue</name>
<fieldLabel>Area Tax Due</fieldLabel>
<dataType>CURRENCY</dataType>
<value>0.0</value>
</field>
<field>
<id>75617</id>
<name>allocatedAmount</name>
<fieldLabel>Allocated Amount</fieldLabel>
<dataType>CURRENCY</dataType>
<value>0.0</value>
</field>
</tableRow>
</table>
<line>
<field>
<id>75612</id>
<name>salesToLocationNotInArea</name>
<lineLabel>Sales Delivered to Locations Not in Any Area</lineLabel>
<dataType>CURRENCY</dataType>
<value>0.0</value>
</field>
</line>
<line>
<field>
<id>75613</id>
<name>netAmountSubjectToLocalTax</name>
<lineLabel>Net Amount Subject to Local Tax</lineLabel>
<dataType>CURRENCY</dataType>
<value>0.0</value>
</field>
</line>
<line>
<field>
<id>75611</id>
<name>transactionsSubjectToLocalTax</name>
<lineLabel>Transactions Subject to Local Tax</lineLabel>
<dataType>CURRENCY</dataType>
<value>0.0</value>
</field>
</line>
<line>
<field>
<id>75614</id>
<name>totalLocalTax</name>
<lineLabel>Total Local Tax</lineLabel>
<dataType>CURRENCY</dataType>
<value>0.0</value>
</field>
</line>
</section>
<section>
<id>75591</id>
<name>preparerInformation</name>
<sectionLabel>Preparer Information</sectionLabel>
<sequence>1</sequence>
<line>
<field>
<id>75592</id>
<name>taxpayerTelephoneNumber</name>
<lineLabel>Taxpayer Telephone Number</lineLabel>
<dataType>TEXT</dataType>
<value/>
</field>
</line>
```

```
<line>
<field>
<id>75593</id>
<name>preparationDate</name>
<lineLabel>Preparation Date</lineLabel>
<dataType>DATE</dataType>
<value/>
</field>
</line>
<line>
<field>
<id>75594</id>
<name>paidPreparerTelephoneNumber</name>
<lineLabel>Paid Preparer Telephone Number</lineLabel>
<dataType>TEXT</dataType>
<value/>
</field>
</line>
<line>
<field>
<id>75595</id>
<name>paidPreparerIdType</name>
<lineLabel>Paid Preparer ID Type</lineLabel>
<dataType>TEXT</dataType>
<value/>
</field>
</line>
<line>
<field>
<id>75596</id>
<name>paidPreparerIdNumber</name>
<lineLabel>Paid Preparer ID Number</lineLabel>
<dataType>TEXT</dataType>
<value/>
</field>
</line>
</section>
</formData>
</documentData>
```

XSLT

This XSLT transforms generic form data using the value in the <name> element as the tag name of the form-specific XML.

For example, it transforms the following fragment:

```
<field>
<name>total</name>
<value>120.00</value>
</field>
```

...into the following single tag: `<total>120.00</total>`

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:msxsl="urn:schemas-microsoft-com:xslt" exclude-result-prefixes="msxsl">

<xsl:output method="xml" indent="yes"/>

<xsl:element name="formType"><xsl:value-of select="formType"/></xsl:element>
<xsl:element name="formCategory"><xsl:value-of select="formCategory"/></xsl:element>
<xsl:element name="documentLocator"><xsl:value-of select="documentLocator"/></xsl:element>
<xsl:template match="customProperties">
<xsl:if test="propertyName" >
<xsl:variable name="propertyName" select="propertyName"/>
<xsl:element name="{$propertyName}">
<xsl:value-of select="value"/>
</xsl:element>
```

```
</xsl:if>
</xsl:template>

<xsl:template match="formData">
<xsl:element name="formID"><xsl:value-of select="id"/></xsl:element>
<xsl:element name="formName"><xsl:value-of select="id"/></xsl:element>
<xsl:apply-templates select="section"/>
</xsl:template>

<xsl:template match="section">
<xsl:variable name="sectionName" select="name" />

<xsl:element name="{$sectionName}">
<xsl:element name="sequence"><xsl:value-of select="sequence"/></xsl:element>
<xsl:apply-templates select="line/field"/>
<xsl:apply-templates select="table"/>
</xsl:element>
</xsl:template>

<xsl:template match="table">
<xsl:variable name="tableName" select="name"/>
<xsl:element name="{$tableName}">
<xsl:apply-templates select="tableRow">
<xsl:with-param name="myTableName" select="$tableName"/>
</xsl:apply-templates>
</xsl:element>
</xsl:template>

<xsl:template match="tableRow">
<xsl:param name="myTableName"></xsl:param>
<xsl:variable name="rowname" select="concat($myTableName,'List')"/>

<xsl:element name="{$rowname}">
<xsl:element name="sequence"><xsl:value-of select="sequence"/></xsl:element>
<xsl:apply-templates select="field"/>
</xsl:element>
</xsl:template>

<xsl:template match="field">
<xsl:variable name="fieldName" select="name" />
<xsl:element name="{$fieldName}">
<xsl:value-of select="value"/>
</xsl:element>
</xsl:template>

<xsl:template match="@* | node()">
<xsl:copy>
<xsl:apply-templates select="@* | node()"/>
</xsl:copy>
</xsl:template>
</xsl:stylesheet>
```

Form-specific XML

Transformation result:

```
<?xml version="1.0" encoding="UTF-8"?><documentData>
<formType>SALESDST2008</formType>
<formCategory>TAXFORM</formCategory>
<currency/>
<formTypeDescription>Sales and Use (w/ Local Distribution) Tax Form</formTypeDescription>
<documentLocator/>
<documentLocatorLabel>Document Locator</documentLocatorLabel>
<CITY>Tacoma</CITY>
<CITY>London</CITY>
<formID>SALESDST2008</formID>
<formName>SALESDST2008</formName>
```

```
<taxpayerInfo>
<sequence>1</sequence>
<filingEndDate>03/31/2012</filingEndDate>
<name>Cooks For a Cause</name>
<idType>EIN</idType>
<idNumber>55-8947563</idNumber>
<country>USA</country>
<address>1 main st.</address>
<address2/>
<city>Midwood</city>
<state>VA</state>
<zipCode>09809</zipCode>
<accountNumber/>
<filingStartDate>01/01/2012</filingStartDate>
</taxpayerInfo>
<lineItems>
<sequence>1</sequence>
<totalGrossSales>120000.0</totalGrossSales>
<transactionsSubjectToLocalTax>0.0</transactionsSubjectToLocalTax>
<purchasesSubjectToTax>2543.0</purchasesSubjectToTax>
<total>122543.0</total>
<totalExemptTransactions>0.0</totalExemptTransactions>
<taxableTransactions>122543.0</taxableTransactions>
<taxRate>10.0</taxRate>
<totalAssessedTaxAmount>12254.3</totalAssessedTaxAmount>
<localTax>0.0</localTax>
<totalTax>12254.3</totalTax>
<firstPrepayment>0.0</firstPrepayment>
<secondPrepayment>0.0</secondPrepayment>
<taxPrepayments>0.0</taxPrepayments>
<remainingTaxDue>12254.3</remainingTaxDue>
<penalty>0.0</penalty>
<interest>0.0</interest>
<totalAmountOwed>12254.3</totalAmountOwed>
</lineItems>
<deductExempt>
<sequence>1</sequence>
<salesToOtherRetailers>0.0</salesToOtherRetailers>
<nonTaxableSales>0.0</nonTaxableSales>
<nonTaxableLabor>0.0</nonTaxableLabor>
<salesToGovernment>0.0</salesToGovernment>
<salesInterstateForeignCommer>0.0</salesInterstateForeignCommer>
<salesTaxInGrossSales>0.0</salesTaxInGrossSales>
<lossesTaxableSales>0.0</lossesTaxableSales>
<lenderLosses>0.0</lenderLosses>
<taxPaidPurchaseResold>0.0</taxPaidPurchaseResold>
<returnedTaxableMerchandise>0.0</returnedTaxableMerchandise>
<cashDiscountsTaxableSales>0.0</cashDiscountsTaxableSales>
<totalFullDeductExempt>0.0</totalFullDeductExempt>
</deductExempt>
<localTaxCompSchedule>
<sequence>1</sequence>
<salesToLocationNotInArea>0.0</salesToLocationNotInArea>
<netAmountSubjectToLocalTax>0.0</netAmountSubjectToLocalTax>
<transactionsSubjectToLocalTax>0.0</transactionsSubjectToLocalTax>
<totalLocalTax>0.0</totalLocalTax>
<compArea>
<compAreaList>
<sequence>1</sequence>
<localAreaName>Denvile</localAreaName>
<localAreaTaxRate>0.05</localAreaTaxRate>
<areaTaxDue>0.0</areaTaxDue>
<allocatedAmount>0.0</allocatedAmount>
</compAreaList>
</compArea>
</localTaxCompSchedule>
<preparerInformation>
<sequence>1</sequence>
```

```
<taxpayerTelephoneNumber/>
<preparationDate/>
<paidPreparerTelephoneNumber/>
<paidPreparerIdType/>
<paidPreparerIdNumber/>
</preparerInformation>
</documentData>
```

Create a Custom Report

The following <documentData> sample is transformed into Report-specific XML using XSLT.

Sample Report Data

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
<documentData>
<formType>2008SALESUSE</formType>
<formTypeDescription>Sales and Use 2008 (with Local Distribution)</formTypeDescription>
<formCategory>TAXFORM</formCategory>
<currency>USD</currency>
<customProperties>
<sequence>1</sequence>
<propertyName>CITY</propertyName>
<propertyDescription>City</propertyDescription>
<value>Tacoma</value>
</customProperties>
<customProperties>
<sequence>2</sequence>
<propertyName>CITY</propertyName>
<propertyDescription>City</propertyDescription>
<value>London</value>
</customProperties>
<documentLocator>TTF-U890625</documentLocator>
<documentLocatorLabel>Document Locator</documentLocatorLabel>
<formData>
<id>82036165155</id>
<name>SALESUSE2008</name>
<formDescription>Sales and Use 2008</formDescription>
<section>
<id>1233</id>
<name>exemption</name>
<sectionLabel>Exemptions</sectionLabel>
<sequence>0</sequence>
<line>
<field>
<id>1234</id>
<name>exemptPrimaryInd</name>
<lineLabel>Primary Taxpayer Exempt</lineLabel>
<dataType>boolean</dataType>
<value>false</value>
</field>
</line>
<line>
<field>
<id>1234</id>
<name>exemptSpouseInd</name>
```

```
<lineLabel>Spouse Exempt</lineLabel>
<dataType>boolean</dataType>
<value>true</value>
</field>
</line>
<line>
<field>
<id>1234</id>
<name>exemptSpouseName</name>
<lineLabel>Exempt Spouse Name</lineLabel>
<dataType>text</dataType>
<value>Mary Smith</value>
</field>
</line>
</section>
<section>
<id>1234</id>
<name>officerComp</name>
<sectionLabel>Officer Compensations</sectionLabel>
<sequence>1</sequence>
<line>
<field>
<id>2344</id>
<name>officeLocation</name>
<lineLabel>Location</lineLabel>
<dataType>text</dataType>
<value>69-01 South Port Dr., Millford VA 80892</value>
</field>
</line>
<line>
<field>
<id>2344</id>
<name>totalCompensation</name>
<lineLabel>Total Compensation</lineLabel>
<dataType>money</dataType>
<value>14700.90</value>
</field>
</line>
<table>
<id>66682</id>
<name>compensations</name>
<tableLabel>Computation</tableLabel>
<tableRow>
<sequence>1</sequence>
<field>
<id>3166</id>
<name>officerLegalName</name>
<fieldLabel>Legal Name</fieldLabel>
<dataType>text</dataType>
<value>Brian Kelly</value>
</field>
<field>
<id>3166</id>
<name>officerSSN</name>
<fieldLabel>SSN</fieldLabel>
<dataType>text</dataType>
<value>981-77-1010</value>
</field>
<field>
<id>2344</id>
<name>salary</name>
<fieldLabel>Salary</fieldLabel>
<dataType>money</dataType>
<value>3507.23</value>
</field>
</tableRow>
<tableRow>
<sequence>2</sequence>
```

```
<field>
<id>3166</id>
<name>officerLegalName</name>
<fieldLabel>Legal Name</fieldLabel>
<dataType>text</dataType>
<value>Pier Morgan</value>
</field>
<field>
<id>3166</id>
<name>officerSSN</name>
<fieldLabel>SSN</fieldLabel>
<dataType>text</dataType>
<value>080-81-5335</value>
</field>
<field>
<id>2344</id>
<name>salary</name>
<fieldLabel>Salary</fieldLabel>
<dataType>money</dataType>
<value>2400.23</value>
</field>
</tableRow>
</table>
</section>
<section>
<id>1234</id>
<name>officerComp</name>
<sectionLabel>Officer Compensations</sectionLabel>
<sequence>2</sequence>
<line>
<field>
<id>2344</id>
<name>officeLocation</name>
<lineLabel>Location</lineLabel>
<dataType>text</dataType>
<value>425 Grove st., New Heaven CT 65010</value>
</field>
</line>
<line>
<field>
<id>2344</id>
<name>totalCompensation</name>
<lineLabel>Total Compensation</lineLabel>
<dataType>money</dataType>
<value>75133.00</value>
</field>
</line>
<table>
<id>66682</id>
<name>compensations</name>
<tableLabel>Computation</tableLabel>
<tableRow>
<sequence>1</sequence>
<field>
<id>3166</id>
<name>officerLegalName</name>
<fieldLabel>Legal Name</fieldLabel>
<dataType>text</dataType>
<value>Wendy Shark</value>
</field>
<field>
<id>3166</id>
<name>officerSSN</name>
<fieldLabel>SSN</fieldLabel>
<dataType>text</dataType>
<value>809-88-2918</value>
</field>
<field>
```

```
<id>2344</id>
<name>salary</name>
<fieldLabel>Salary</fieldLabel>
<dataType>money</dataType>
<value>4108.66</value>
</field>
</tableRow>
<tableRow>
<sequence>2</sequence>
<field>
<id>3166</id>
<name>officerLegalName</name>
<fieldLabel>Legal Name</fieldLabel>
<dataType>text</dataType>
<value>Bob Marley</value>
</field>
<field>
<id>3166</id>
<name>officerSSN</name>
<fieldLabel>SSN</fieldLabel>
<dataType>text</dataType>
<value>421-11-3222</value>
</field>
<field>
<id>2344</id>
<name>salary</name>
<fieldLabel>Salary</fieldLabel>
<dataType>money</dataType>
<value>1000.99</value>
</field>
</tableRow>
</table>
</section>
</formData>
</documentData>
```

XSLT Transform

```
<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:msxsl="urn:schemas-microsoft-com:xslt" exclude-result-prefixes="msxsl">

<xsl:output method="xml" indent="yes"/>

<xsl:element name="formType"><xsl:value-of select="formType"/></xsl:element>
<xsl:element name="formCategory"><xsl:value-of select="formCategory"/></xsl:element>
<xsl:element name="documentLocator"><xsl:value-of select="documentLocator"/></xsl:element>
<xsl:template match="customProperties">
<xsl:variable name="propertyName" select="propertyName"/>
<xsl:element name="{$propertyName}">
<xsl:value-of select="value"/>
</xsl:element>
</xsl:template>

<xsl:template match="formData">
<xsl:element name="formID"><xsl:value-of select="id"/></xsl:element>
<xsl:element name="formName"><xsl:value-of select="name"/></xsl:element>
<xsl:apply-templates select="section"/>
</xsl:template>

<xsl:template match="section">
<xsl:variable name="sectionName" select="name" />
```

```
<xsl:element name="{$sectionName}">
<xsl:element name="sequence"><xsl:value-of select="sequence"/></xsl:element>
<xsl:apply-templates select="line/field"/>
<xsl:apply-templates select="table"/>
</xsl:element>
</xsl:template>

<xsl:template match="table">
<xsl:variable name="tableName" select="name"/>
<xsl:element name="{$tableName}">
<xsl:apply-templates select="tableRow">
<xsl:with-param name="myTableName" select="$tableName"/>
</xsl:apply-templates>
</xsl:element>
</xsl:template>

<xsl:template match="tableRow">
<xsl:param name="myTableName"></xsl:param>
<xsl:variable name="rowname" select="concat($myTableName,'List')"/>

<xsl:element name="{$rowname}">
<xsl:element name="sequence"><xsl:value-of select="sequence"/></xsl:element>
<xsl:apply-templates select="field"/>
</xsl:element>
</xsl:template>

<xsl:template match="field">
<xsl:variable name="fieldName" select="name" />
<xsl:element name="{$fieldName}">
<xsl:value-of select="value"/>
</xsl:element>
</xsl:template>

<xsl:template match="@* | node()">
<xsl:copy>
<xsl:apply-templates select="@* | node()"/>
</xsl:copy>
</xsl:template>
</xsl:stylesheet>
```

Report-specific XML

The result of the transformation:

```
<?xml version="1.0" encoding="UTF-8"?><documentData>
<formType>SALESDST2008</formType>
<formCategory>TAXFORM</formCategory>
<currency/>
<formTypeDescription>Sales and Use (w/ Local Distribution) Tax Form</formTypeDescription>
<documentLocator/>
<documentLocatorLabel>Document Locator</documentLocatorLabel>
<CITY>Tacoma</CITY>
<CITY>London</CITY>
<formID>SALESDST2008</formID>
<formName>SALESDST2008</formName>
<taxpayerInfo>
<sequence>1</sequence>
<filingEndDate>03/31/2012</filingEndDate>
<name>Cooks For a Cause</name>
<idType>EIN</idType>
<idNumber>55-8947563</idNumber>
<country>USA</country>
<address>1 main st.</address>
<address2/>
<city>Midwood</city>
```

```
<state>VA</state>
<zipCode>09809</zipCode>
<accountNumber/>
<filingStartDate>01/01/2012</filingStartDate>
</taxpayerInfo>
<lineItems>
<sequence>1</sequence>
<totalGrossSales>120000.0</totalGrossSales>
<transactionsSubjectToLocalTax>0.0</transactionsSubjectToLocalTax>
<purchasesSubjectToTax>2543.0</purchasesSubjectToTax>
<total>122543.0</total>
<totalExemptTransactions>0.0</totalExemptTransactions>
<taxableTransactions>122543.0</taxableTransactions>
<taxRate>10.0</taxRate>
<totalAssessedTaxAmount>12254.3</totalAssessedTaxAmount>
<localTax>0.0</localTax>
<totalTax>12254.3</totalTax>
<firstPrepayment>0.0</firstPrepayment>
<secondPrepayment>0.0</secondPrepayment>
<taxPrepayments>0.0</taxPrepayments>
<remainingTaxDue>12254.3</remainingTaxDue>
<penalty>0.0</penalty>
<interest>0.0</interest>
<totalAmountOwed>12254.3</totalAmountOwed>
</lineItems>
<deductExempt>
<sequence>1</sequence>
<salesToOtherRetailers>0.0</salesToOtherRetailers>
<nonTaxableSales>0.0</nonTaxableSales>
<nonTaxableLabor>0.0</nonTaxableLabor>
<salesToGovernment>0.0</salesToGovernment>
<salesInterstateForeignCommer>0.0</salesInterstateForeignCommer>
<salesTaxInGrossSales>0.0</salesTaxInGrossSales>
<lossesTaxableSales>0.0</lossesTaxableSales>
<lenderLosses>0.0</lenderLosses>
<taxPaidPurchaseResold>0.0</taxPaidPurchaseResold>
<returnedTaxableMerchandise>0.0</returnedTaxableMerchandise>
<cashDiscountsTaxableSales>0.0</cashDiscountsTaxableSales>
<totalFullDeductExempt>0.0</totalFullDeductExempt>
</deductExempt>
<localTaxCompSchedule>
<sequence>1</sequence>
<salesToLocationNotInArea>0.0</salesToLocationNotInArea>
<netAmountSubjectToLocalTax>0.0</netAmountSubjectToLocalTax>
<transactionsSubjectToLocalTax>0.0</transactionsSubjectToLocalTax>
<totalLocalTax>0.0</totalLocalTax>
<compArea>
<compAreaList>
<sequence>1</sequence>
<localAreaName>Denvile</localAreaName>
<localAreaTaxRate>0.05</localAreaTaxRate>
<areaTaxDue>0.0</areaTaxDue>
<allocatedAmount>0.0</allocatedAmount>
</compAreaList>
</compArea>
</localTaxCompSchedule>
<preparerInformation>
<sequence>1</sequence>
<taxpayerTelephoneNumber/>
<preparationDate/>
<paidPreparerTelephoneNumber/>
<paidPreparerIdType/>
<paidPreparerIdNumber/>
</preparerInformation>
</documentData>
```

Creating a Form-Specific Report

Steps to create a form-specific report:

1. Build/define the form-specific XML sample. For example, fill and submit the form online and use the SOA Enterprise Management Console to get the sample generic form data from the BPEL Process instance XML. Apply the XSL to transform the generic data into form-specific XML.
2. Login to the BI Publisher Enterprise Console, navigate to a Catalog, and locate the directory in which the product reports are located (see the *PSRMSS Installation Guide* for additional details).
3. Create a new Data Model. Use the form data XML created above to add a data set. It can also be used as sample data.
4. Establish a naming convention for Report Layout names. It should be coordinated with the BPEL process custom logic for layout name derivation. For example, the Form Type code may be used as the Layout name.
5. Create a new Report and add a Layout using the native BI Publisher's Layout Editor or Microsoft Word (for RTF templates).