Oracle® Automotive Trading Partner Toolkit User's Guide

User Guide, Release 11i

March 2000

Part No. A83734_01

ORACLE

Oracle® Automotive Trading Partner Toolkit User's Guide, Release 11i

Part No. A83734_01

Primary Author: Brian Gause

Contributing Authors: Nikhil Parekh, Srinivasa Munagala

Contributors: Sai Addepalli, Anil Verma, David Reitan

Contents

Customers 3-1

Forms and Utilities 4-1

Send Us Your Comments

Oracle Automotive Trading Partner Toolkit User's Guide, Release 11i

Part No. A83734_01

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

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

If you find any errors or have any other suggestions for improvement, please indicate the chapter, section, and page number (if available). You can send comments to us in the following ways:

- E-mail - Automotive Trading Partner Toolkit@us.oracle.com
- FAX - telephone number. Attn: Oracle Automotive Trading Partner Toolkit
- Postal service: 3op
  Oracle Corporation
  Oracle Automotive Trading Partner Toolkit Documentation
  500 Oracle Parkway, 3op6
  Redwood Shores, CA 94065
  USA

If you would like a reply, please give your name, address, and telephone number below.

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

Preface

Audience for This Guide

Welcome to Release 11*i* of the Oracle® Automotive Trading Partner Toolkit User's Guide.

This guide assumes you have a working knowledge of the following:

- The principles and customary practices of your business area.
- Oracle® Release Management

  If you have never used Oracle® Release Management, we suggest you attend one or more of the Oracle® Order Management training classes available through Oracle University.
- Oracle® Automotive Trading Partner Toolkit
- The Oracle Applications graphical user interface.

  To learn more about the Oracle Applications graphical user interface, read the *Oracle Applications User Guide.*

See Other Information Sources for more information about Oracle Applications product information.

How To Use This Guide

This guide contains the information you need to understand and use Oracle® Automotive Trading Partner Toolkit.

This preface explains how this user guide is organized and introduces other sources of information that can help you. This guide contains the following chapters:

- Chapter 1 gives an overview of Oracle® Automotive

  Note: Implementation information and procedures are included in this chapter.

- Chapter 2 explains the step-by-step procedures for Layer Providers, including instructions for planning and creating trading partner layers.
- Chapter 3 explains the step-by-step procedures for customers, including how to develop site-specific customizations and activate trading partner layers.
- Chapter 4 describes the layer workbench and the utilities for interfacing with the Trading Partner Toolkit Repository.
- The appendices provide you with complete navigation paths to all windows in Oracle® Automotive Trading Partner Toolkit, restrictions on imported code and, for layer developers, sample driver files.

Finding Out What's New

From the HTML help window for Oracle® Automotive Trading Partner Toolkit, choose the section that describes new features or what's new from the expandable menu. This section describes:

- New features in 11*i*. This information is updated for each new release of Oracle® Automotive Trading Partner Toolkit.
- Information about any features that were not yet available when this user guide was printed. For example, if your system administrator has installed software from a mini pack as an upgrade, this document describes the new features.

Other Information Sources

You can choose from many sources of information, including online documentation, training, and support services, to increase your knowledge and understanding of Oracle® Automotive Trading Partner Toolkit.

If this guide refers you to other Oracle Applications documentation, use only the Release 11*i* versions of those guides unless otherwise specified.

Online Documentation

All Oracle Applications documentation is available online (HTML and PDF). The technical reference guides are available in paper format only. Note that the HTML documentation is translated into over twenty languages.

The HTML version of this guide is optimized for on screen reading, and you can use it to follow hypertext links for easy access to other HTML guides in the library. When you have an HTML window open, you can use the features on the left side of the window to navigate freely throughout all Oracle Applications documentation.

- You can use the Search feature to search by words or phrases.
- You can use the expandable menu to search for topics in the menu structure we provide. The Library option on the menu expands to show all Oracle Applications HTML documentation.

You can view HTML help in the following ways:

- From an application window, use the help icon or the help menu to open a new Web browser and display help about that window.
- Use the documentation CD.
- Use a URL provided by your system administrator.

Your HTML help may contain information that was not available when this guide was printed.

Related User Guides

Oracle® Automotive Trading Partner Toolkit shares business and setup information with other Oracle Applications products. Therefore, you may want to refer to other user guides when you set up and use Oracle® Automotive Trading Partner Toolkit.

You can read the guides online by choosing Library from the expandable menu on your HTML help window, by reading from the Oracle Applications Document Library CD included in your media pack, or by using a Web browser with a URL that your system administrator provides.

If you require printed guides, you can purchase them from the Oracle store at http://oraclestore.oracle.com.

User Guides Related to All Products

Oracle Applications User Guide

This guide explains how to navigate the system, enter data, and query information, and introduces other basic features of the GUI available with this release of Oracle® Automotive Trading Partner Toolkit (and any other Oracle Applications product).

You can also access this user guide online by choosing *Getting Started and Using Oracle Applications* from the Oracle Applications help system.

Oracle Alert User Guide

Use this guide to define periodic and event alerts that monitor the status of your Oracle Applications data.

Oracle Applications Implementation Wizard User Guide

If you are implementing more than one Oracle product, you can use the Oracle Applications Implementation Wizard to coordinate your setup activities. This guide describes how to use the wizard.

Oracle Applications Developer's Guide

This guide contains the coding standards followed by the Oracle Applications development staff. It describes the Oracle Application Object Library components needed to implement the Oracle Applications user interface described in the *Oracle Applications User Interface Standards*. It also provides information to help you build your custom Oracle Developer forms so that they integrate with Oracle Applications.

Oracle Applications User Interface Standards

This guide contains the user interface (UI) standards followed by the Oracle Applications development staff. It describes the UI for the Oracle Applications products and how to apply this UI to the design of an application built by using Oracle Forms.

User Guides Related to This Product

Oracle Applications Demonstration User's Guide

This guide documents the functional storyline and product flows for Vision Enterprises, a fictional manufacturer of personal computers products and services. As well as including product overviews, the book contains detailed discussions and examples across each of the major product flows. Tables, illustrations, and charts summarize key flows and data elements.

Oracle e-Commerce Gateway User's Guide

This guide describes how Oracle e-Commerce Gateway provides a means to conduct business with trading partners via Electronic Data Interchange (EDI). Data files are exchanged in a standard format to minimize manual effort, speed data processing and ensure accuracy.

Oracle Release Management User's Guide

This manual describes how to manage high volume electronic demand by continually incorporating your customers demand into your order and planning processes. By explaining how to validate, archive, manage and reconcile incoming planning, shipping and production sequence schedules with updates to sales orders and forecasts, it enables you to electronically collaborate with your customers to more accurately manage demand. It also describes how to plan, create and manage trading partner layers for trading partner specific customizations.

Reference Manuals

Oracle Technical Reference Manuals

Each technical reference manual contains database diagrams and a detailed description of database tables, forms, reports, and programs for a specific Oracle Applications product. This information helps you convert data from your existing applications, integrate Oracle Applications data with non-Oracle applications, and write custom reports for Oracle Applications products.

You can order a technical reference manual for any Oracle Applications product you have licensed.

Oracle Release Management Implementation Manual

This manual describes the setup and implementation of the Oracle Applications used for the Oracle Automotive solution, including Oracle Release Management and Oracle Automotive Trading Partner Toolkit.

Oracle Manufacturing and Distribution Open Interfaces Manual

This manual contains up-to-date information about integrating with other Oracle Manufacturing applications and with your other systems. This documentation includes open interfaces found in Oracle Manufacturing.

Oracle Applications Message Reference Manual

This manual describes all Oracle Applications messages. This manual is available in HTML format on the documentation CD-ROM for Release 11i.

Oracle Applications Flexfields Guide

This guide provides flexfields planning, setup, and reference information for the Oracle® Automotive Trading Partner Toolkit implementation team, as well as for users responsible for the ongoing maintenance of Oracle Applications product data. This guide also provides information on creating custom reports on flexfields data.

Installation and System Administration Guides

Oracle Applications Concepts

This guide provides an introduction to the concepts, features, technology stack, architecture, and terminology for Oracle Applications Release 11*i*. It provides a useful first book to read before an installation of Oracle Applications. This guide also introduces the concepts behind, and major issues, for Applications-wide features such as Business Intelligence (BIS), languages and character sets, and self-service applications.

Installing Oracle Applications

This guide provides instructions for managing the installation of Oracle Applications products. In Release 11*i*, much of the installation process is handled using Oracle One-Hour Install, which minimizes the time it takes to install Oracle Applications and the Oracle 8*i* Server technology stack by automating many of the required steps. This guide contains instructions for using Oracle One-Hour Install and lists the tasks you need to perform to finish your installation. You should use

this guide in conjunction with individual product user guides and implementation guides.

Upgrading Oracle Applications

Refer to this guide if you are upgrading your Oracle Applications Release 10.7 or Release 11.0 products to Release 11*i*. This guide describes the upgrade process in general and lists database upgrade and product-specific upgrade tasks. You must be at either Release 10.7 (NCA, SmartClient, or character mode) or Release 11.0 to upgrade to Release 11*i*. You cannot upgrade to Release 11*i* directly from releases prior to 10.7.

Using the AD Utilities

Use this guide to help you run the various AD utilities, such as AutoInstall, AutoPatch, AD Administration, AD Controller, Relink, and others. It contains how-to steps, screenshots, and other information that you need to run the AD utilities.

Oracle Applications Product Update Notes

Use this guide as a reference if you are responsible for upgrading an installation of Oracle Applications. It provides a history of the changes to individual Oracle Applications products between Release 11.0 and Release 11*i*. It includes new features and enhancements and changes made to database objects, profile options, and seed data for this interval.

Oracle Applications System Administrator's Guide

This guide provides planning and reference information for the Oracle Applications System Administrator. It contains information on how to define security, customize menus and online help, and manage processing.

Oracle Self-Service Purchasing Implementation Manual

This manual describes how to set up Oracle Self-Service Purchasing. Self-Service Purchasing enables employees to requisition items through a self-service, Web interface.

Oracle Workflow Guide

This guide explains how to define new workflow business processes as well as customize existing Oracle Applications-embedded workflow processes. You also

use this guide to complete the setup steps necessary for any Oracle Applications product that includes workflow-enabled processes.

Training and Support

Training

We offer a complete set of training courses to help you and your staff master Oracle Applications. We can help you develop a training plan that provides thorough training for both your project team and your end users. We will work with you to organize courses appropriate to your job or area of responsibility.

Training professionals can show you how to plan your training throughout the implementation process so that the right amount of information is delivered to key people when they need it the most. You can attend courses at any one of our many Educational Centers, or you can arrange for our trainers to teach at your facility. We also offer Net classes, where training is delivered over the Internet, and many multimedia-based courses on CD. In addition, we can tailor standard courses or develop custom courses to meet your needs.

Support

From on-site support to central support, our team of experienced professionals provides the help and information you need to keep Oracle® Automotive Trading Partner Toolkit working for you. This team includes your Technical Representative, Account Manager, and Oracle's large staff of consultants and support specialists with expertise in your business area, managing an Oracle server, and your hardware and software environment.

Do Not Use Database Tools to Modify Oracle Applications Data

We STRONGLY RECOMMEND that you never use SQL*Plus, Oracle Data Browser, database triggers, or any other tool to modify Oracle Applications tables, unless we tell you to do so in our guides.

Oracle provides powerful tools you can use to create, store, change, retrieve, and maintain information in an Oracle database. But if you use Oracle tools such as SQL*Plus to modify Oracle Applications data, you risk destroying the integrity of your data and you lose the ability to audit changes to your data.

Because Oracle Applications tables are interrelated, any change you make using an Oracle Applications form can update many tables at once. But when you modify Oracle Applications data using anything other than Oracle Applications forms, you

might change a row in one table without making corresponding changes in related tables. If your tables get out of synchronization with each other, you risk retrieving erroneous information and you risk unpredictable results throughout Oracle Applications.

When you use Oracle Applications forms to modify your data, Oracle Applications automatically checks that your changes are valid. Oracle Applications also keeps track of who changes information. But, if you enter information into database tables using database tools, you may store invalid information. You also lose the ability to track who has changed your information because SQL*Plus and other database tools do not keep a record of changes.

About Oracle

Oracle Corporation develops and markets an integrated line of software products for database management, applications development, decision support and office automation, as well as Oracle Applications. Oracle Applications provides the E-business Suite, a fully integrated suite of more than 70 software modules for financial management, Internet procurement, business intelligence, supply chain management, manufacturing, project systems, human resources and sales and service management.

Oracle products are available for mainframes, minicomputers, personal computers, network computers, and personal digital assistants, enabling organizations to integrate different computers, different operating systems, different networks, and even different database management systems, into a single, unified computing and information resource.

Oracle is the world's leading supplier of software for information management, and the world's second largest software company. Oracle offers its database, tools, and application products, along with related consulting, education and support services, in over 145 countries around the world.

Your Feedback

Thank you for using Oracle® Automotive Trading Partner Toolkit and this user guide.

We value your comments and feedback. This guide contains a Reader's Comment Form you can use to explain what you like or dislike about Oracle® Automotive Trading Partner Toolkit or this user guide. Mail your comments to the following address or call us directly at (650) 506-7000.

Oracle Applications Documentation Manager
Oracle Corporation

500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Or, send electronic mail to **appsdoc@us.oracle.com**.

1

Trading Partner Architecture

Overview of Trading Partner Toolkit

Oracle Automotive Trading Partner Toolkit provides a mechanism and tools to develop, add and maintain trading partner specific customizations separate from base products. It allows you to rapidly capture, develop and execute a solution to meet trading partner requirements or industry standards.

Oracle developers register customizable program units in a central repository. Layer developers then provide trading partner specific extensions by writing code with a different processing behavior for the program units published in the repository. Next, these new program units can be gathered into sets, called layers, and seamlessly merged with base code for Oracle Automotive so that trading partner customizations are automatically recognized and processed by the software. Additionally, Oracle Automotive Trading Partner Toolkit ensures that these layers are maintained independently from the underlying program units, and not overwritten by future bug fixes or patches as part of the Oracle base product maintenance.

Trading Partner Toolkit Mechanism

Trading partner specific processing requires the application to change the transaction flow to execute custom code in place of generic code for the processing of specific trading partners. The Trading Partner Toolkit supports this by introducing two special types of program units:

- TPS Program Units: Trading partner selector (TPS) program units derive the trading partner being processed in the transaction from the context information.
- TP Program Units: Trading partner program units insulate generic code from custom code. Calls to customizable generic code and custom code are always made through this TP program unit. In other words, customizable generic code and custom code are never explicitly called within the product except by TP program units. This program units first calls the TPS program unit to determine the trading partner being processed in the transaction, then calls either custom code or generic code depending on the trading partner being processed.

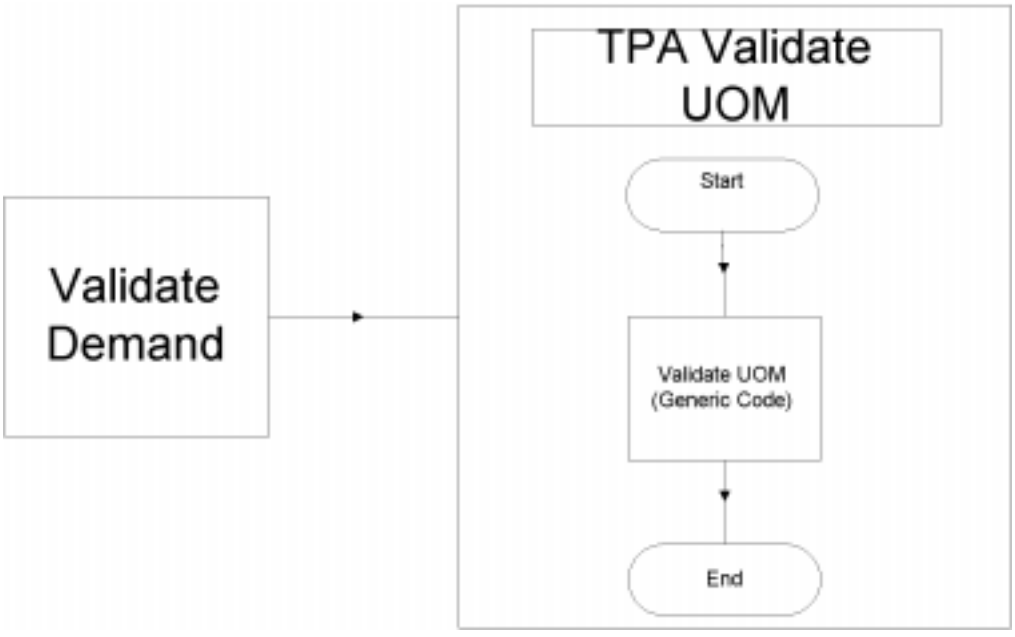
  The trading partner architecture maintains a registry of customizable program units, custom code, TP program units, TPS program units and associations among them. The TP program unit is always generated by the architecture using this registry.

The following example illustrates the basic mechanism of trading partner layers. In the three figures that follow:

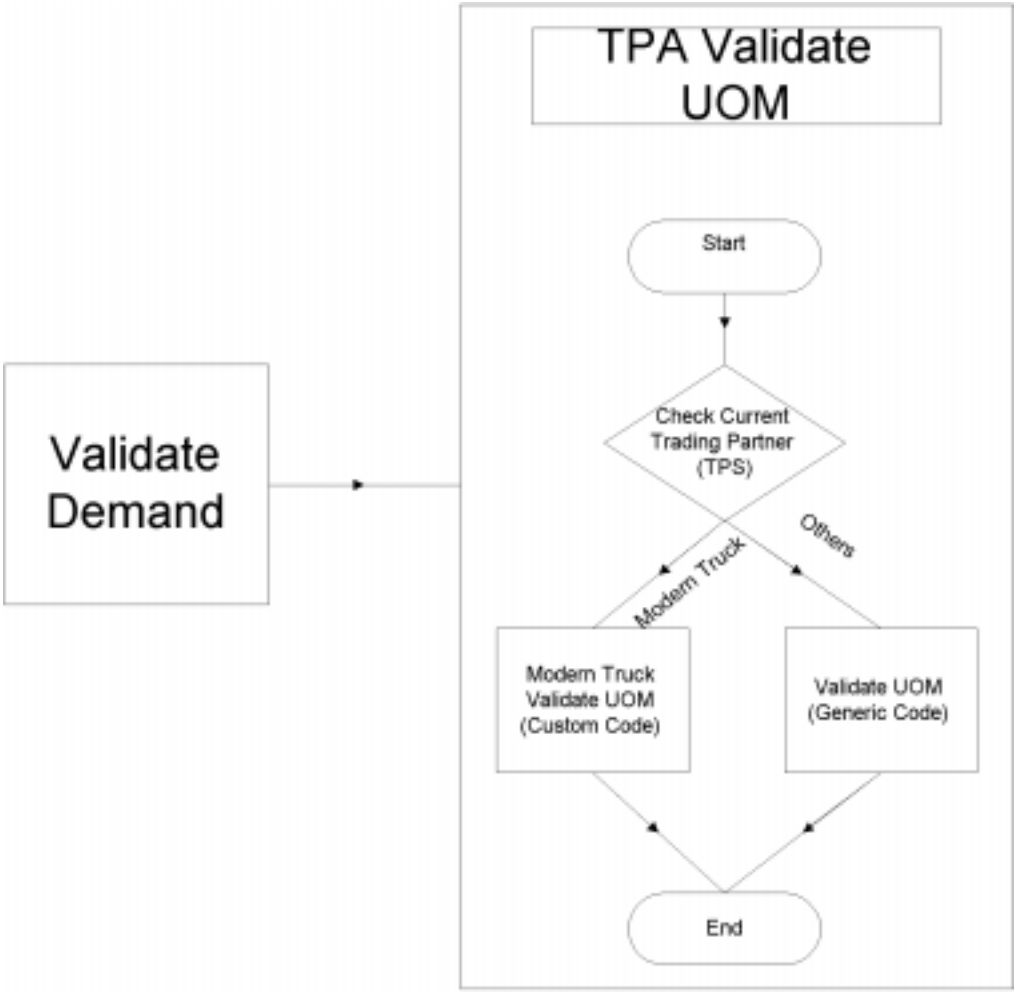
- ValidateUOM is the customizable generic program unit.
- ValidateDemand calls ValidateUOM.
- TPAValidateUOM is the TP program unit for ValidateUOM.
- ModernTruckValidateUOM is the custom program unit developed for the trading partner “Modern Truck”.

The following figure illustrates the logic of the transaction flow before the base product is enabled for the Trading Partner Toolkit:

The following figure represents the logic of the transaction flow after the base product is enabled, but before any customizations are developed for the Trading Partner Toolkit:



The following figure illustrates the logic of the transaction flow after customizations have been built and integrated into the base product:



Functional Tools

Layer developers and customers can use a number of tools to respond to trading partner mandates. This Trading Partner Toolkit encapsulates a strategy for describing when and how to use each tool, and a process to manage layers from multiple sources.

In order to implement a trading partner mandate, you must carefully analyze the business requirements and determine the most appropriate options. For example, a trading partner's requirements for netting schedules can be addressed by using processing rules in Oracle Release Management, but a specific data element must be captured through a trading partner flexfield and processed through a trading partner layer.

Translators

Translators offer robust mapping tools that let you map trading partner specific transaction formats into Oracle e-Commerce Gateway. Many of these mapping tools provide extensive data transformation capabilities, and have easy-to-use interfaces that let you manage changes quickly and easily.

Changes to a trading partner file format may simply require a change in a translator mapping. Careful analysis of the trading partner change can identify the requirements and indicate whether any additional changes need to be made in Oracle Applications.

E-Commerce Gateway Transaction Templates

Oracle e-Commerce Gateway offers an extensive selection of features to help you transform trading partner data. With Oracle e-Commerce Gateway, you can generate unique inbound and outbound templates for transactions to accommodate additional trading partner data, such as flexfields.

Translators can perform data derivation, but cannot source trading partner data from the application. Using Oracle e-Commerce Gateway, you can capture and publish this additional data to a translator. To avoid inconsistencies, you must also ensure that each version of a transaction template is closely tied to a version of the translator map.

E-Commerce Gateway Code Conversions

The Oracle e-Commerce Gateway lets you define code conversion values for data elements, so you can transform trading partner data into your own application data. You can establish trading partner values and automatically transform inbound and outbound data to reflect this conversion.

Application Setup Data

Within each application, you can leverage flexible setup data to accommodate trading partner requirements. For example, in Oracle Release Management you can

define trading partner processing rules by customer / address or item. These rules include how to consume demand, interpret new schedules, etc.

Trading Partner Flexfields

As part of Oracle Automotive Trading Partner Toolkit, you can add new attributes in Oracle Release Management, Order Management and Shipping Execution to accommodate trading partner specific attributes. These products have added additional columns in key tables, reserved for trading partner flexfields. You can seed appropriate values for each trading partner and use Oracle Applications flexfield technology to view these attributes from the appropriate forms.

Trading Partner Layers

In addition to transforming and adding data, you can also process and derive data based on specific trading partner mandates. In Oracle Release Management and Oracle Shipping, you can adapt "published" code so that it processes according to specific requirements without impacting the processing for other trading partners. You can create custom procedures, then identify and register a group of procedures as a layer. In this way, you can build trading partner libraries in trading partner-enabled forms in Oracle Release Management and Oracle Shipping Execution.

Layers can also be turned on and off to identify problems with specific trading partner processing. By linking multiple procedures across products, into one layer, you can easily control all processing for a specific trading partner or location.

The installation mechanism also simplifies the process of patching a trading partner layer and the specific code, without disrupting other processing. Patches include all required seed data, e.g. flexfields, that the layer requires for trading partner processing.

Workflow

You can accommodate additional trading partner mandates by examining and modifying the workflow information in Oracle Order Management and Oracle Shipping Execution.

Processing Customizations

For products not included in the initial release of Oracle Trading Partner Toolkit, you can use conventional methods to adapt standard processing to a trading partner's requirements. These customizations must be managed individually and are not included as part of the patching strategy.

Report Customizations

To accommodate trading partner specific reporting, you can build customized reports and link them to transactions using the trading partner architecture. For example, if the trading partner “Modern Truck” requires a packing slip in a different format, you can develop a custom packing slip report and link it to the ship-confirm transaction. When items are delivered to Modern Truck, this custom report will generate the packing slip. Custom reports can reference published code to leverage your customizations, but these custom reports must be managed individually, and are not included as part of the patching strategy.

2

Layer Developers

The procedures listed here are specific to Layer Developers using the Oracle Automotive Trading Partner Architecture to create layers. For the procedures specific to customers, see Chapter 3, Customer Procedures.

Overview of Layer Developer Procedures

Oracle Automotive Trading Partner Toolkit provides a mechanism and tools to develop, add and maintain trading partner specific customizations separate from base products. It allows you to rapidly capture, develop and execute a solution to meet trading partner requirements or industry standards.

Oracle developers register customizable program units in a central repository. Layer developers then provide trading partner specific extensions by writing code with a different processing behavior for the program units published in the repository. Next, these new program units can be gathered into sets, called layers, and seamlessly merged with base code for Oracle Automotive so that trading partner customizations are automatically recognized and processed by the software. Additionally, Oracle Automotive Trading Partner Toolkit ensures that these layers are maintained independently from the underlying program units, and not overwritten by future bug fixes or patches as part of the Oracle base product maintenance.

Prerequisites

Before proceeding, you must set the value for the following profile option to ensure accurate processing:

- VEA: Layer Provider

See the Oracle System Administrator's User's Guide for more detailed instructions on setting profile options.

Note: To proceed through the Layer Developer procedures listed here, set the profile option **VEA:Layer Provider** to the layer provider code assigned to you by Oracle.

Trading Partner Specific Processing

The following steps describe how to develop and create trading partner layers for Oracle Trading Partner Toolkit.

Step	Description
1	Upload TP Metadata File from Oracle
2	Perform Functional Analysis of Customization Requirements
3	Plan and Develop Trading Partner Specific Code
4	Create a Trading Partner Layer
5	Insert TP tags in Layer Code
6	Import Layer Code
7	Associate Trading Partner Layers with Base Layers
8	Generate TP Code
9	Test the Trading Partner Layers
10	Create Trading Partner Layer Licenses
11	Generate TP Metadata file for the Trading Partner Layer
12	Package the Trading Partner Layer

For steps describing how to create context-sensitive structures for flexfields to handle trading partner specific attributes, see: Trading Partner Specific Attributes on page 2-23.

Step 1: Upload TP Metadata File from Oracle

Apply the release and patch drivers for all required products from Oracle. This will populate the Trading Partner Toolkit repository with the base layer, which consists of code developed by Oracle development teams and independent of trading partners.

Step 2: Perform Functional Analysis of Customization Requirements

Analyze the customization requirements and business feasibility issues to identify the trading partner layers that need to be developed as well as any additional trading partner specific attributes required. You must have a good understanding of Oracle base product before performing functional analysis. You can:

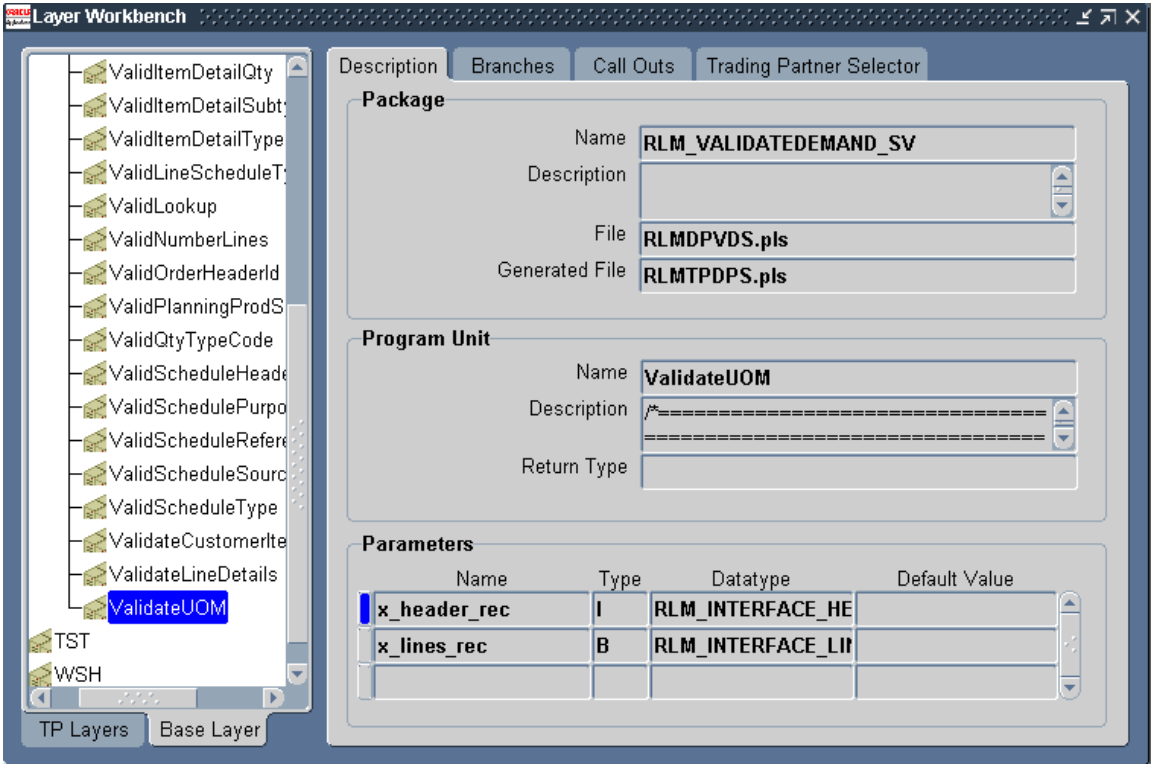
- examine Technical Reference Manuals for data model details
- examine this User's Guide for functionality and features
- examine the base layer using the Layer Workbench for details about published program units.

Step 3: Plan and Develop Trading Partner Specific Code

Examine the program units in the base layer using the Layer Workbench to identify the program units which need trading partner specific code.

◗◗ To examine base layer program units:

1. Through Automotive Manager>Trading Partner Toolkit>Layer Workbench, navigate to the Layer Workbench.



2. Select the Base Layer tab on the left side and the Description tab on the right side of this window.
3. Expand the items in the base layer tree to examine the program units.
4. Select a program unit or package on the left side to view the name, description and further details on the right side of the window.
5. Take note of the program units that need site-specific code.

Create a PL/SQL package to hold the mirror trading partner layer program units for each of the base layer program units that need trading partner specific code. For each of the packages you create, ensure that the following naming prefixes for names of the package and its files are used.

- The package name should begin with the layer provider code and underscore character. For example, if your layer provider code is LP2, you might use package name LP2_MT_TPA_SV.

- The package file name should begin with the layer provider code. For example, if your layer provider code is LP1, a valid package specification file would be LP2DPVDS.pls, and a valid body file would be LP2DPVDB.pls.

See Also

Layer Workbench

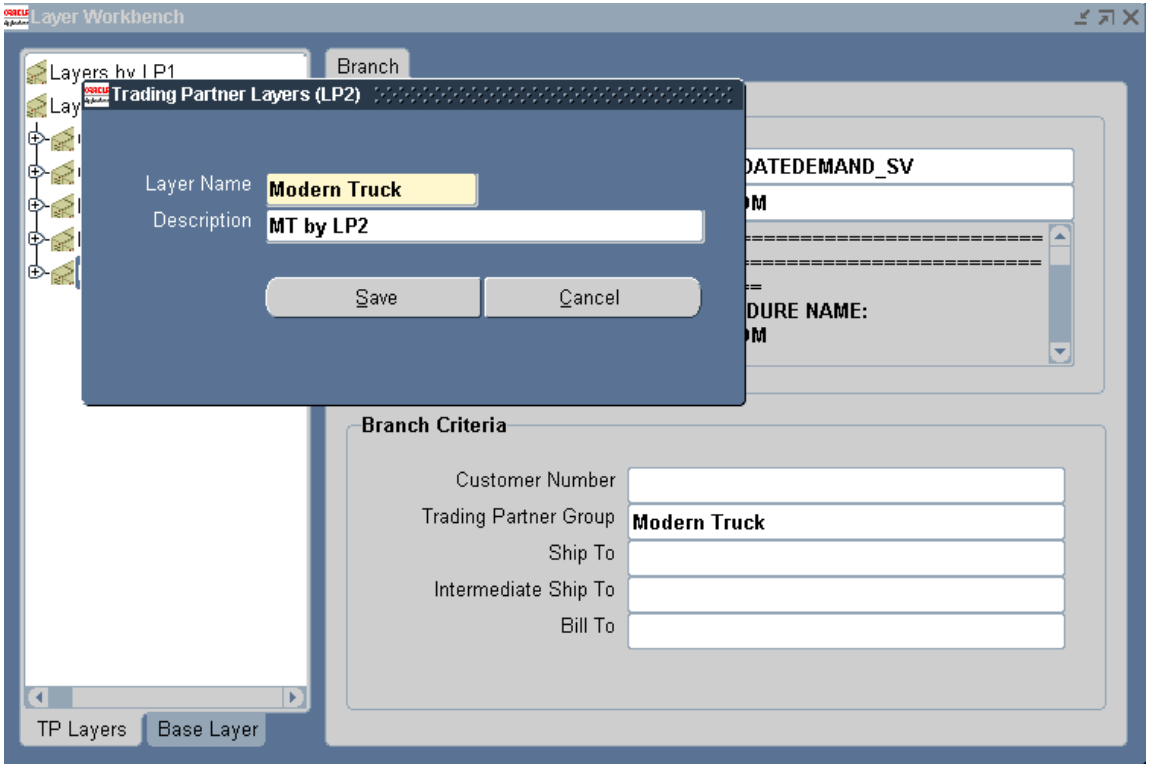
Step 4: Create a Trading Partner Layer

A trading partner layer is the collection of PL/SQL packages which contain trading partner specific code. These PL/SQL program units will perform trading partner specific processing or validations which you have designed.

Identify the trading partner for which trading partner specific code is to be provided, then create a trading partner layer using the Layer Workbench.

To create a trading partner layer:

1. Through Automotive Manager>Trading Partner Toolkit>Development on the main menu, navigate to the Layer Workbench.



2. Select the TP Layers tab on the left side of the window.
3. Right-click on the mouse to select New Layer from the pop-up menu.
4. Enter the name and a description of the new trading partner layer.
5. Save your work.

See Also

Layer Workbench

Step 5: Insert TP tags in Layer Code

For each of the packages that contain mirror program units corresponding to the base layer program units, follow the steps below to add trading partner layer tags.

◗◗ To insert TP tags into the layer code:

For each of the packages containing mirror program units corresponding to the base layer program units, add a package level tag anywhere after the "CREATE OR REPLACE" statement and before any uncommented package contents.

- <TPA_LAYER=Name>

  where Name is the name of the trading partner layer you created in Step 4.

  For example, if the validateUOM program unit is published and is visible in the base layer and a Modern Truck Layer program unit called MTValidateUOM is developed in Package LP2_TPA_MT_SV with file LP2MTVDS.pls, the specification file with the TP tags would like this:

```
CREATE or REPLACE PACKAGE LP2_TPA_MT_SV as
/* $Header: LP2MTVDS.pls 115.13 99/10/12 16:12:09 ship $ */
/* =================== LP2_TPA_MT_SV ================ */
--<TPA_LAYER=Modern Truck>
--Specifies that this package contains
--Modern Truck specific code
...
...
...
PROCEDURE MTValidateUOM
           (
             x_header_rec  IN RLM_INTERFACE_HEADERS%ROWTYPE,
             x_lines_rec IN OUT RLM_INTERFACE_LINES%ROWTYPE
           )
;
...
...
...
...
END MTValidateUOM;

END LP2_TPA_MT_SV;
```

Step 6: Import Layer Code

After you have inserted the TP tags into the code, import the trading partner layer code into the repository. The trading partner layer program units are then visible in the TP Layers tree of the Layer Workbench.

◗◗ To import trading partner layer code:

1. Set up the Oracle Applications environment. Verify that the environment variable TWO_TASK points to the correct database instance.
2. Run the import utility with the following command line

   java oracle.apps.vea.main GWYUID=<value of environment variable GWYUID> TWO_TASK=<value of environment variable TWO_TASK> FNDNAM=<value of environment variable FNDNAM> USERID=<Userid> FUNCTION=IMPORT APPLICATION=<App> FILE=<File Name>

   where:

   - USERID is the username and password, e.g. GLOBAL/GLOBAL93.
   - APPLICATION is the application shortname, i.e. either RLM or WSH.
   - FILE is the PL/SQL specification file name ending with S.pls. This file should be present in the same directory from which this utility is run.
3. Verify that no errors are displayed by the import utility.
4. Through Automotive Manager>Trading Partner Toolkit>Development on the main menu, navigate to the Layer Workbench.
5. Verify that these imported program units are visible under the TP Layer tree of the Layer Workbench.

See Also

Layer Workbench

Import Utility

Step 7: Associate Trading Partner Layers with Base Layers

Trading partner layers are associated with the base layer through a link between a trading partner layer program unit and a base layer program unit called a branch. Mirror program units developed in a trading partner layer for each of the base layer program units that need trading partner specific code are linked to their corresponding base layer program units through the branches you specify using the Layer Workbench.

The following three steps are necessary for associating trading partner layers with base layers:

- Choose a trading partner selector for the base layer program unit, or

Develop and import trading partner selectors.

- Create a branch for the trading partner program unit.
- Verify the new branch.

Choose a Trading Partner Selector

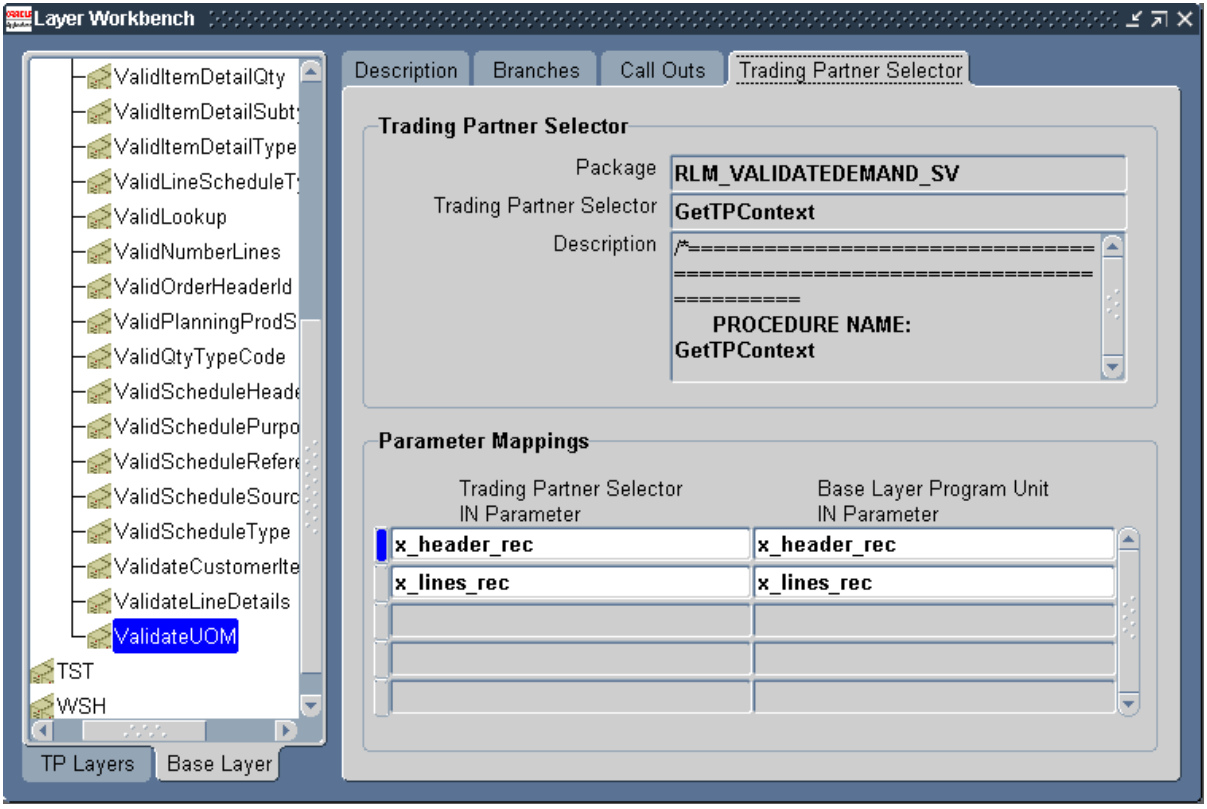
To begin the process of associating trading partner layers with base layers, you must choose a trading partner selector. A trading partner selector is a PL/SQL procedure which derives trading partner information from transaction data. The trading partner selector can have any number of parameters, but it must have exactly five VARCHAR2 parameters (OUT/IN OUT) as specified below:

Trading Partner Group Code	x_tp_group_code
Customer Number	x_customer_number
Ship-To EDI Location Code	x_ship_to_ece_locn_code
Intermediate Ship-To EDI Location Code	x_inter_ship_to_ece_locn_code
Bill-To EDI Location Code	x_bill_to_ece_locn_code

You must specify a trading partner selector for a base layer program unit before you can build branches on the base layer program unit. The trading partner selector is invoked by the generated code and the output trading partner information determines at runtime which trading partner specific branch to execute.

◗◗ To choose a trading partner selector for the base layer program unit:

1. Identify the corresponding base layer and trading partner program units you want to link through a branch.
2. Navigate to the Layer Workbench through Automotive Manager>Trading Partner Toolkit>Development on the menu.



3. Select the Base Layer tab on the left of the Layer Workbench.
4. Select the appropriate base layer program unit in the hierarchical tree on the left.
5. Select the Trading Partner Selector tab on the right.
6. If the trading partner selector already exists, proceed to step nine. If there is no current trading partner selector, choose the package from the list of values.
7. Select the appropriate program unit from the list of values.
8. Enter a description of this trading partner selector.
9. In the Parameter Mappings region of the Layer Workbench, enter the Trading Partner Selector IN Parameters and corresponding base layer program unit IN Parameters that you want to link.

10. Save your work.

Develop and Import Trading Partner Selectors

You may need to create separate trading partner selectors. For each of the base layer program units, there should be at least one trading partner selector with matching IN parameters.

For example, if there are two published program units validateUOM and validateItem both taking Schedule Line as an IN parameter, one trading partner selector that takes Schedule Line as IN parameter would suffice. If there are any other published program units that do not have Schedule Line as IN parameter, more trading partner selector procedures that handle the IN parameters of those other published program units are required.

Follow the steps below to add tags for all the trading partner selector procedures you have developed and import them into the repository.

1. For each developed trading partner selector program unit, add a program unit level tag at the end of its specification

2. The tag for designating a program unit as a trading partner selector is

```
--<TPA_TPS>
```

3. For example, if the getTPContext program unit of the package LP2_TPS_SV with specification file LP2PTPSS.pls is to be designated as a trading partner selector, the specification file with the TP tags would like this:

```
CREATE or REPLACE PACKAGE LP2_TPS_SV as
/* $Header: LP2PTPSS.pls 115.13 99/10/12 16:12:09 ship $*/
/* ==================== LP2_TPS_SV ================= */
...
PROCEDURE getTPContext
          (
..
..
);
--<TPA_TPS>
--Specifies that this procedure is a Trading Partner Selector
..
..

END LP2_TPS_SV;
```

4. Set up the Oracle Applications environment. Verify that the environment variable TWO_TASK points to the correct database instance.
5. Run the import utility with the following command line

   java oracle.apps.vea.main GWYUID=<value of environment variable GWYUID> TWO_TASK=<value of environment variable TWO_TASK> FNDNAM=<value of environment variable FNDNAM> USERID=<Userid> FUNCTION=IMPORT APPLICATION=<App> FILE=<File Name>

   where:

- USERID is the username and password, e.g. GLOBAL/GLOBAL93.
- APPLICATION is the application shortname, i.e. either RLM or WSH.
- FILE is the PL/SQL specification file name ending with S.pls. The file should be present in the current directory from which this utility is run.

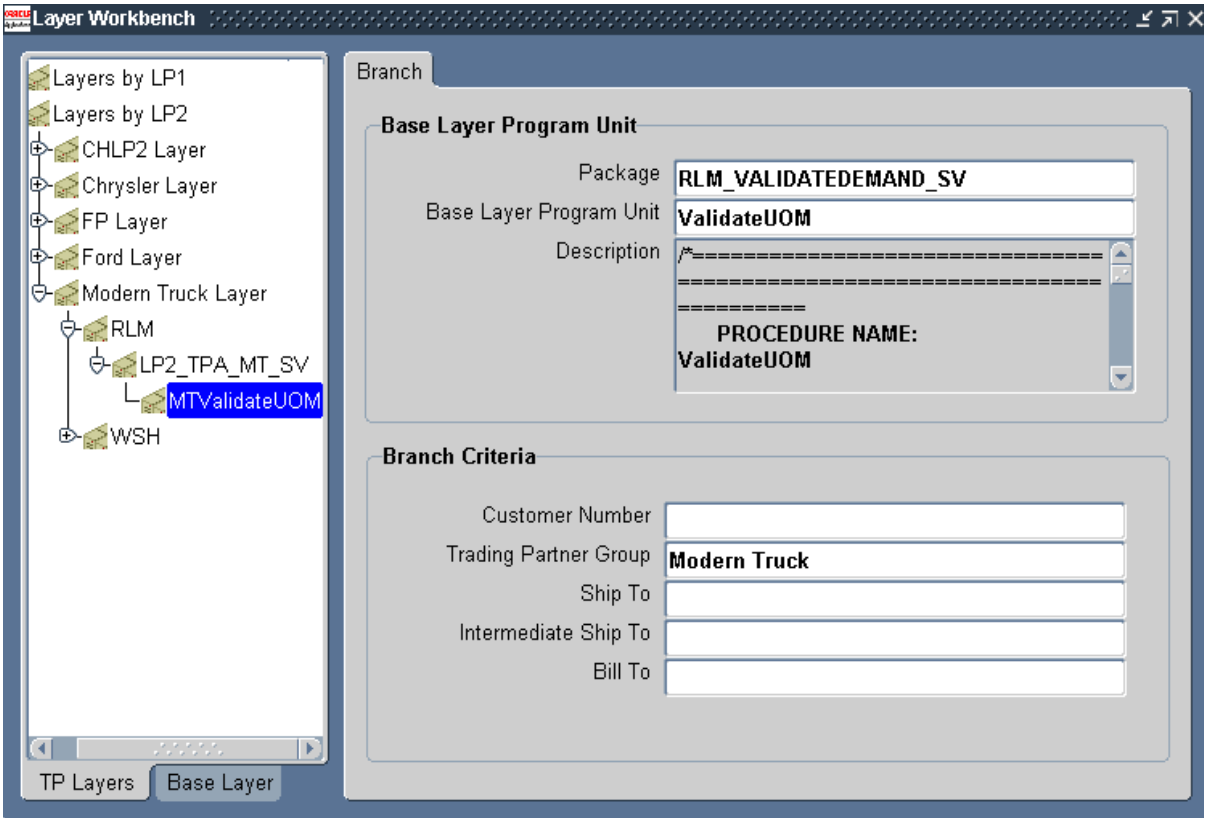
6. Verify that no errors are displayed by the import utility and that imported program units are visible under TP Layer tree of the Layer Workbench.

Create a Branch

After you have chosen a TPS for the base layer program unit, you must next create a branch for the trading partner layer program unit. This will connect the base layer to the trading partner layer.

◗◗ To create a branch for the trading partner layer program unit:

1. Through Automotive Manager>Trading Partner Toolkit>Development on the main menu, navigate to the Layer Workbench.



2. Select the TP Layers tab on the left of the Layer Workbench.
3. Expand the items in the Trading Partner Layer tree to find the trading partner layer program unit of the appropriate branch.
4. Select the program unit to view the corresponding branch information on the right side.
5. Choose the appropriate trading partner layer program unit of the branch.
6. On the right, select the package and base layer program unit from the list of values.
7. Specify the Branch Criteria by entering appropriate values for Customer Number, Trading Partner Group, Ship To, Intermediate Ship To, and Bill To fields.

For example, a branch in the Modern Truck Layer might have a trading partner Group value of “MT” and a Ship To value of “Modern Truck-Cleveland”.

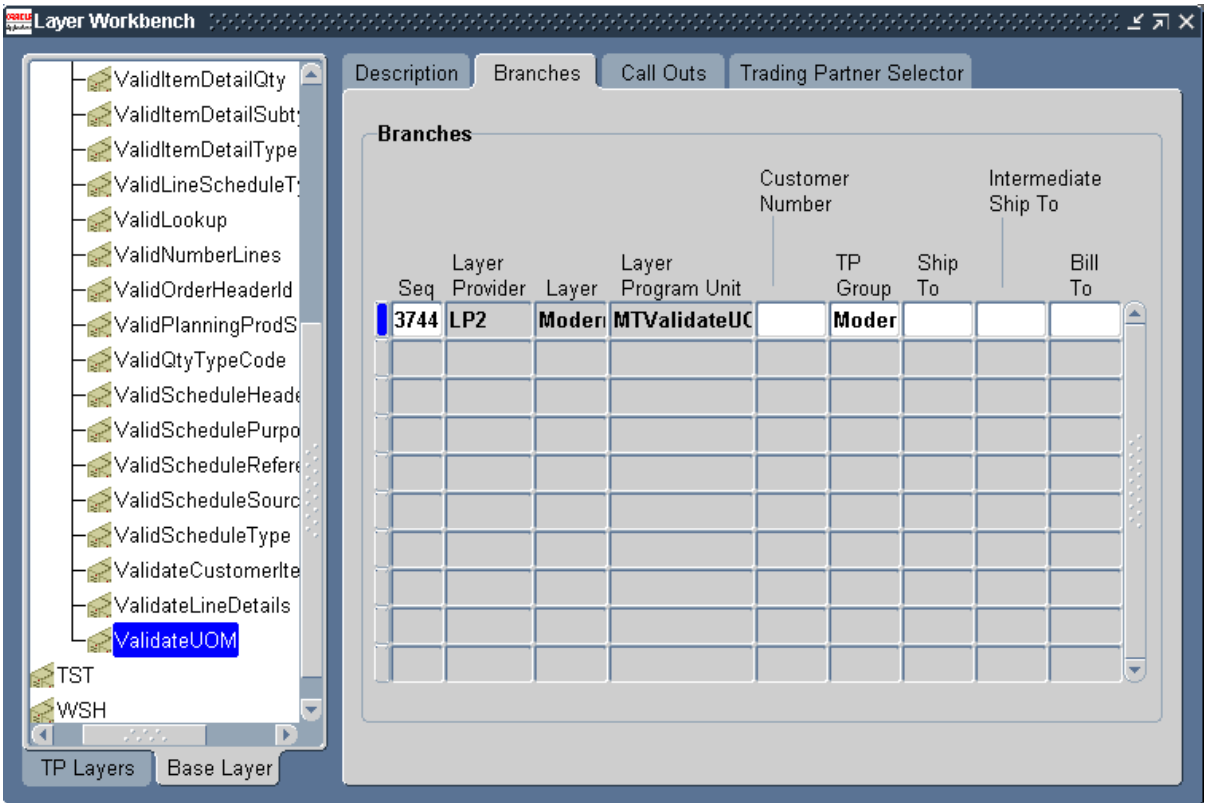
8. Save your work.

Inspect the Branch

After creating a branch to link the trading partner layer to the base layer, you must inspect your work to verify that the branch works as intended.

To verify the new branch:

1. Through Automotive Manager>Trading Partner Toolkit>Development on the menu, navigate to the Layer Workbench.



2. Select the Base Layer tab on the left of the window.

3. Expand the items in the base layer tree to find the appropriate base layer program unit of the branch. Select the corresponding program unit to query the branch information on the right side.
4. Select the Branches tab on the right of the window. You will see all branches associated with the current base layer program unit.
5. Verify that the branch you created is listed in the table on the right and that all data is correct. The sequence number of each branch determines the priority of the branch within a base layer program unit. The lower the sequence number, the higher the priority. Branch criteria will always be evaluated in the order of sequence number of the branch.
6. If necessary, you can update the sequence at this time. Also, if you want to update branch criteria fields, including Customer Number, Trading Partner Group, Ship To, Intermediate Ship To, and Bill To, do so at this time.
7. Save your work.

You have now successfully created a branch between a trading partner layer program unit and a base layer program unit. To create additional branches, repeat these steps.

See Also

Layer Workbench

Step 8: Generate TP code

After you have created trading partner layers in the previous steps, you must generate the TP code. Follow the steps below to run the generate utility and create the generated package files.

Note: For any generated package, you need to compile only package body, you should not compile package specification.

◗◗ To generate the TP code from the Trading Partner Toolkit repository:

1. Set up the Oracle applications environment. Verify that environment variable TWO_TASK points to the correct database instance.
2. Run the generate utility with the following command line

   java oracle.apps.vea.main GWYUID=<value of environment variable GWYUID> TWO_TASK=<value of environment variable TWO_TASK>

FNDNAM=<value of environment variable FNDNAM> USERID=<Userid> FUNCTION=GENERATE APPLICATION=<App> FILE=<File Name>

where:

- USERID is the username and password e.g. GLOBAL/GLOBAL93.
- APPLICATION is the application shortname, i.e. either RLM or WSH.
- FILE is the PL/SQL specification file name of a TP package ending with S.pls. For Release Management, this will be the file name of the common TP Package "RLMTPDPS.pls".

3. Verify that the specification and body files are created in the current directory.

See Also

Layer Workbench

Generate Utility

Step 9: Test the Trading Partner Layers

Compile the generated packages and the packages created for the trading partner layer in your testing database and ensure that the desired functionality is achieved. You can test the functionality by activating and deactivating the trading partner layers in the Layer Workbench.

See Also

Activate a Trading Partner Layer

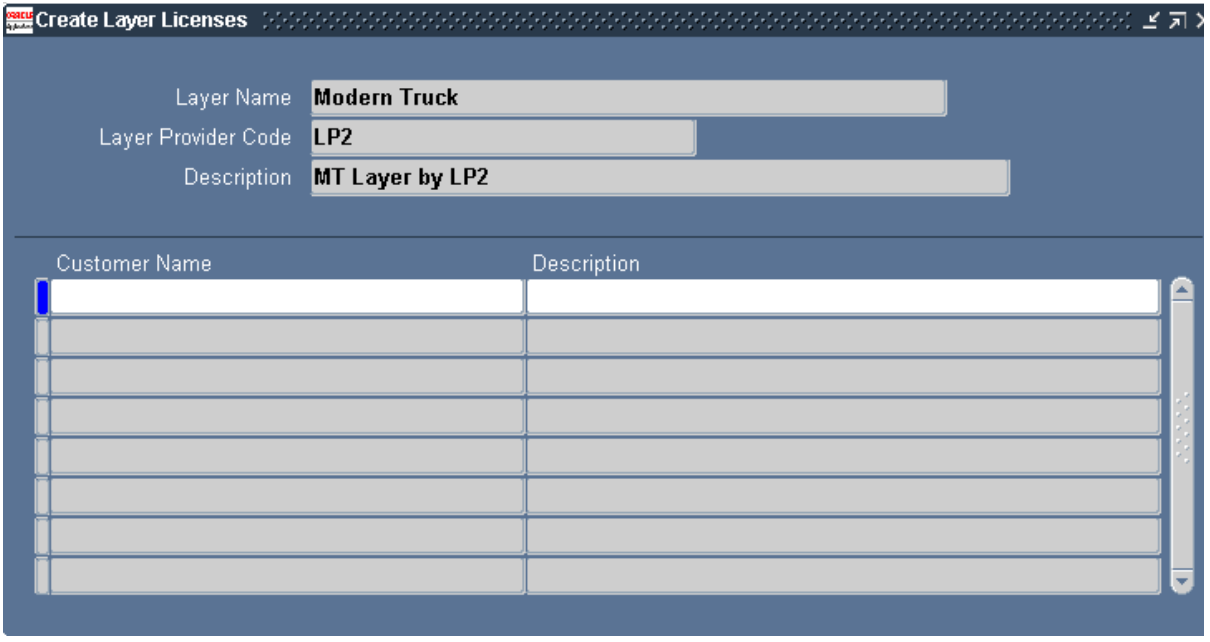
Step 10: Create Trading Partner Layer Licenses

Create trading partner layer licenses for each customer and trading partner layer combination by using the Layer Workbench. The Create Layer Licenses window in this form allows you to create and modify layer licenses for particular trading partner layers. When you select a layer name, all current licenses are listed in the fields below.

Customers licensed to the user-selected trading partner layer name are displayed in this window. For each trading partner layer, customer names and descriptions are listed in the lower half of the window.

To create a trading partner layer license:

1. Through Automotive Manager>Trading Partner Toolkit>Development on the menu, navigate to the Layer Workbench.
2. Select TP Layers Tab on the left side.
3. Expand the nodes in the tree and select the trading partner layer of interest.
4. Right-click on the mouse to invoke the popup menu on the TP Layers tree.
5. Select Trading Partner Layer Licenses in the popup menu to open the Trading Partner Layer Licenses window with the selected trading partner layer in the master block at the top.



6. Create a layer license in the detailed block at the bottom for each of the customers you want to license to this trading partner layer.
   - Enter the name of the customer and descriptive text for each customer licensed to use the layer.
7. Save your changes.

Note: To modify an existing license in the Create Layer Licenses window, select the customer name and make modifications to the description. To remove a license, simply delete the record.

See Also

Layer Workbench

Create Layer Licenses

Activate a Trading Partner Layer

Step 11: Generate TP metadata file for the Trading Partner Layer

Use FNDLOAD to generate the trading partner metadata file from Trading Partner Toolkit repository. This metadata file must be included in all patches.

- Set up the environment.
- Execute the following command:

  FNDLOAD userid/password 0 Y DOWNLOAD $VEA_TOP/admin/import/vea.lct <metadata file name> TPA [APP_SHORT_NAME=app_short_name] [LAYER_NAME=layer_name]

Note: Depending on your environment, the latest version of vea.lct may reside in $VEA_TOP/admin/import/vea.lct or in $VEA_TOP/patch/115/import/vea.lct. Please check with your system administrator and use the correct location in the syntax above.

The following table lists the modes in which TP metadata can be extracted from the repository using FNDLOAD.

Application Short Name specified on command line	Layer Name specified on command line	Extracted Information	Suggested Filename Convention
No	No	All layers in all applications	<lpc>.ldt
Yes	No	All layers in the specified application	<lpc>_<asn>.ldt

Application Short Name specified on command line	Layer Name specified on command line	Extracted Information	Suggested Filename Convention
No	Yes	Specified layer across all applications	<lpc>_<ln>.ldt
Yes	Yes	Specified layer in the specified application	<lpc>_<asn>_<ln>.ldt
Legend:	lpc - Layer Provider Code; asn - Application Short Name; ln - Layer Name		

Step 12: Package the Trading Partner Layer

Now that you have created and tested the trading partner layer, you must complete the process by identifying and creating driver files to ship with the layer.

1. Identify all the files required for the trading partner layer. This includes the following:
   - all trading partner specific processing packages imported under the trading partner layer
   - other packages referenced by the imported trading partner specific packages
   - TP metadata files created for each of the relevant products
   - all driver files which must be created for this trading partner layer
2. Create the following driver files and a readme.txt file:
   - a copy driver file listing all the files identified as part of the trading partner layer,
   - a database driver file which specifies how these files are applied in the database,
   - a generate driver file,
   - a readme.txt file which contains details of changes made and any special instruction for install the patch.
3. All the files shipped to customers should have the following identification line in them:

```
/* $Header: <File> <Version> <Date Time> ship $ */
```

If you are using a source control mechanism to store the files, make sure that the above "ident" information is placed by the source control system. Otherwise, you have to manually place the "ident" line and increment the version information each time you ship. Version information follows the format 115.x where x is the version under release 11.5 (R11I).

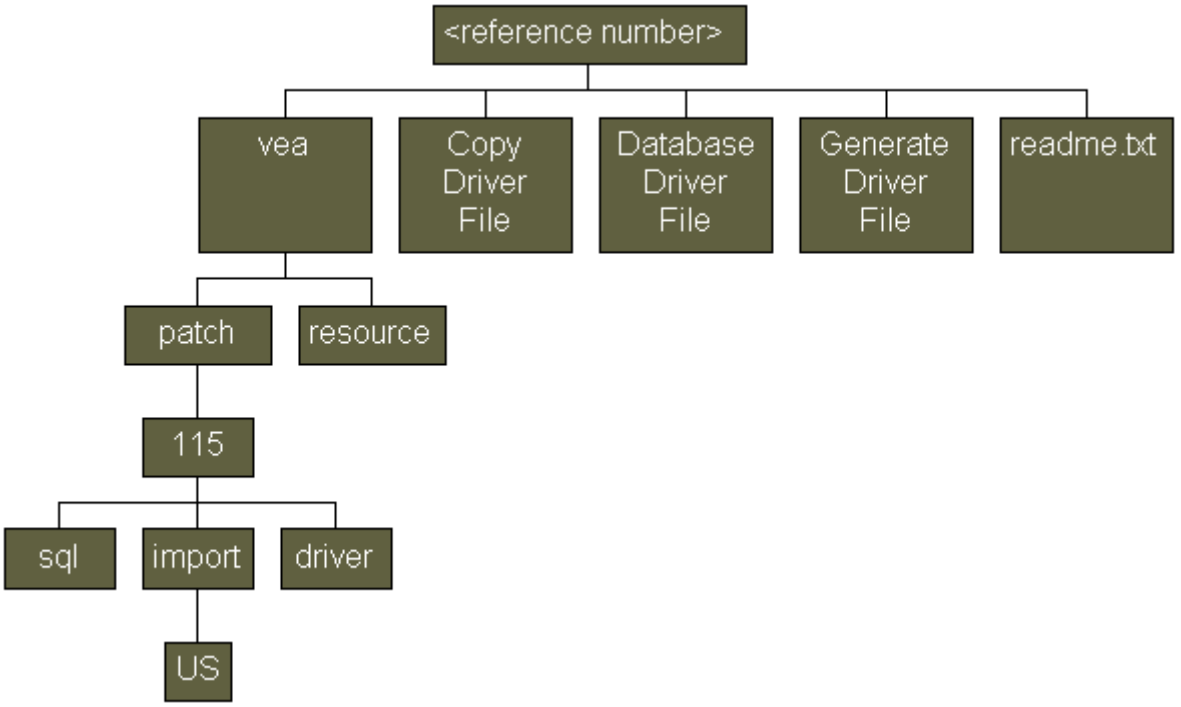
An example ident line is:

```
/* $Header: LP2MTVDS.pls 115.13 99/10/12 16:12:09 ship $ */
```

Note: The above example illustrates an identification line for PL/SQL files and, hence, includes /* .. */ style comments. For the syntax of driver files, you should exclude these comment markers.

Please refer to Appendix B for a sample of driver files

4. Create a stage area:
   - each patch must be linked to a unique reference number
   - each reference number should begin with the layer provider code, e.g. LP1000001
   - create a directory with the name as reference number, referred as the stage area TOP
   - create a subdirectory "vea" under the stage area TOP, referred as the product TOP or Application TOP
5. Organize all files to be shipped under the stage area as shown in the following figure:



Organize your files under the stage area as follows

- copy all the driver files and readme.txt under the stage area top.
- copy all PL/SQL package specification and body (*.pls) files under <reference_number>/vea/patch/115/sql
- copy all PL/SQL library (*.pll) files under <reference_number>/vea/resource
- copy all metadata (*.ldt) files under <reference_number>/vea/patch/115/import/US
- copy database driver file under <reference_number>/vea/patch/115/driver also.

Create a zip file for the stage area as follows:

- include all subfolders
- verify that the zip archive contains relative path information for all subfolders under stage area top

- Follow the conventions below when naming the zip file:

  p<reference number>_<release number>.zip

  for example, plp14_1150.zip

  where:

  - lp14 is reference number, and
  - 1150 is release number for R11*i*

Trading Partner Specific Attributes

Oracle Release Management, Order Management and Shipping Execution now include additional columns in key tables, reserved for trading partner flexfields. Relevant forms and transactions have visibility to these flexfields. As a layer developer, you can create context-sensitive structures for these flexfields to handle trading partner specific attributes, where the context is the trading partner.

Step	Description
1	Create Context Field Values
2	Create Segments for Context Field Values
3	Package Trading Partner Flex Seed Data

Step 1: Create Context Field Values

You can create a number of different sets of segments for a flexfield depending on the value of the context field, which can be defined to refer to a field in the parent form of the flexfield. The context field allows you to define different sets of segments for each distinct value of the corresponding field in the parent form. The context field in a trading partner flexfield refers to the hidden field corresponding to the TP_ATTRIBUTE_CATEGORY column. In this way, you can create different sets of additional data elements for each trading partner.

Create a context field value for each trading partner for which there are additional data elements in the form or table. You can create new context field values for any additional trading partners when the need arises. You can then create segments for all the context field values as described in the next section. The segments created for a specific context field value (corresponding to a trading partner) will appear in the flexfields window exactly when the trading partner selector field has the value of the specific trading partner.

For detailed instructions on creating new context field values, see the Oracle Applications Flexfields Guide.

See Also

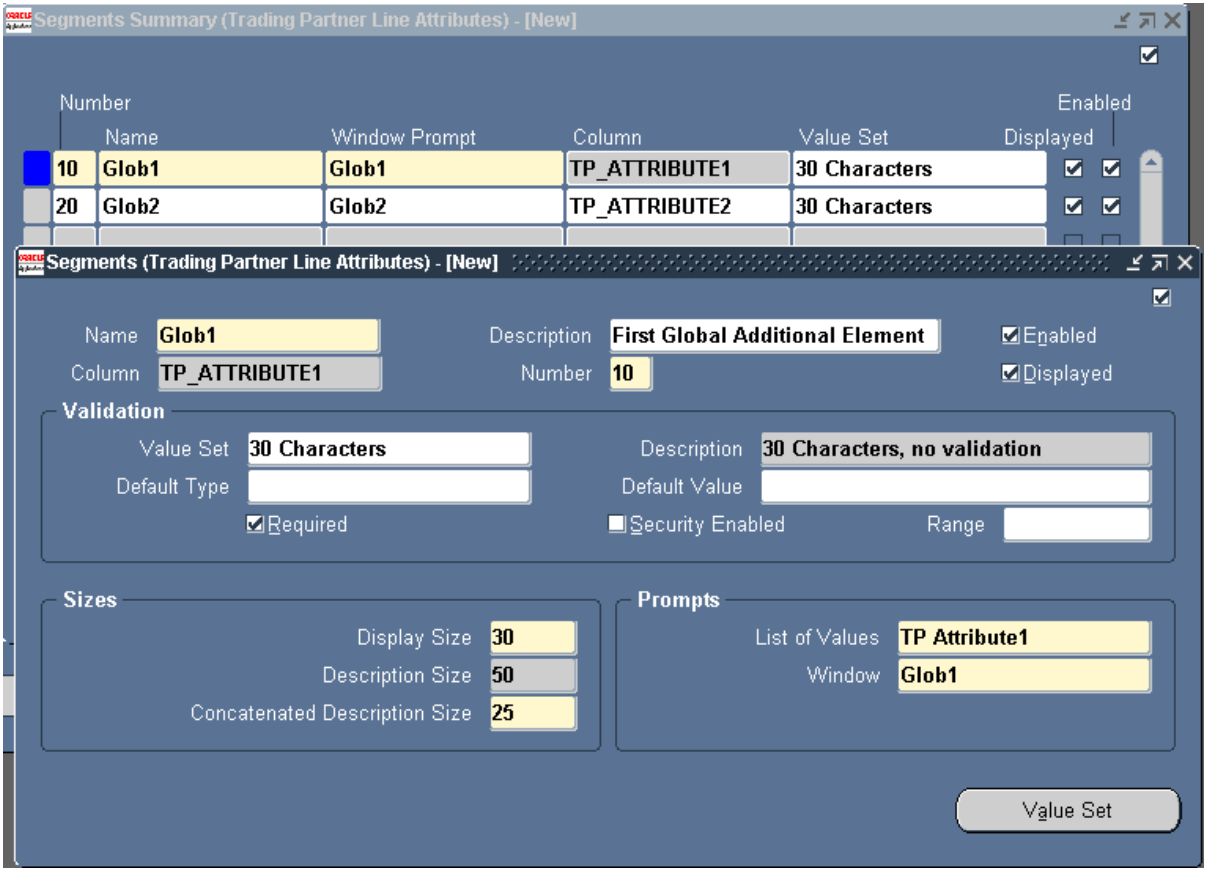
An Example with Context Field Values

Step 2: Create Segments for Context Field Values

Follow the steps below for each segment you want to create under context field value, including the global data elements context field value.

◗◗ To create a segment for a context field value:

1. Through Application Developer>Flexfields>Descriptive on the menu, open the Segments window.



2. Query the Trading Partner Descriptive Flexfield you want to extend.
3. Enter name and window prompt in the Segments window. The window prompt is what appears as the prompt for the segment in the flexfields window.
4. Select a column from the list of values. You can use any attribute column not used by other segments in the same context field value and by the segments in the global data elements context field value. You can also reuse any attribute column used by a segment in a context field value other than global data elements.
5. Select or create a value set to use. This value set specifies the list of valid values for the segment.
6. Open the segment to enter additional information.

7. Enter a default type. Default type is either a constant or another segment or a field in the parent form or a sql statement, etc.
8. Enter a default value. Default value is the actual constant or the name of the segment or the name of the form field or the sql statement, etc.
9. Enter any relevant additional information in the form like the Display or Description sizes or Prompts.
10. Save your work.

See Also

An Example with Context Field Values

Step 3: Package Trading Partner Flex Seed Data

1. Extract the Flexfields metadata using FNDLOAD.

- Set up the Oracle Applications environment.
- Execute the following command:

```
FNDLOAD userid/password 0 Y DOWNLOAD $FND_TOP/admin/import/afffload.lct
<metadata file name>  DESC_FLEX  [APPLICATION_SHORT_NAME=app_short_name]
[DESCRIPTIVE_FLEXFIELD_NAME=descriptive_flexfied_name]
```

Note: Depending on your environment, the latest version of file vea.lct may reside in $FND_TOP/admin/import/vea.lct OR $FND_TOP/patch/115/import/vea.lct. Please check with your system administrator and use the correct location in the syntax above.

2. Identify the files to be packaged.
3. Create driver files including all the above files.

An Example with Context Field Values

This example shows how to handle additional data elements which can vary by trading partner. In a particular form, two additional data elements, e.g. Glob1 and Glob2, are needed for all trading partners, one additional data element, e.g. MTSeg1, for trading partner group MT and three additional data elements, e.g BGSeg1, BGSeg2 and BGSeg3, for trading partner group BG.

◗◗ To extend the trading partner flexfield to handle additional data elements:

1. Through Application Developer>Flexfields>Descriptive on the menu, open the Segments window.
2. Under the Global Data Elements context field value, create segments Glob1 and Glob2 using attribute columns TP_ATTRIBUTE1 and TP_ATTRIBUTE2, respectively.
3. Create a Context Field Value “MT”.
4. Under the Context Field value “MT”, create segment MTSeg1 using attribute column TP_ATTRIBUTE3. You can use any attribute column not already in use for the Global Data Elements segments.
5. Create a Context Field Value “BG”.
6. Under the Context Field value “BG”, create segments BGSeg1, BGSeg2, and BGSeg3 using attribute columns TP_ATTRIBUTE3, TP_ATTRIBUTE4, and TP_ATTRIBUTE5 respectively.

> **Note:** The attribute column TP_ATTRIBUTE3 can be used both for MTSeg1 and BGSeg1, as only one of these segments could be used for a context field value at a time.

After you have set up the segments as described above, the flexfields window will display the following fields for different trading partners:

Trading Partner Selector Field Value	Fields Displayed
MT	Glob1, Glob2, ChSeg1
BG	Glob1, Glob2, BGSeg1, BGSeg2, BGSeg3
<all others>	Glob1, Glob2

7. Save your work.

For more information on Flexfields and creating segments, see the Oracle Applications Flexfields Guide.

3

Customers

The procedures listed here are specific to customers using the Oracle Automotive Trading Partner Architecture. For the procedures specific to Layer Developers, see Chapter 2, Layer Developer Procedures.

Overview of Customer Procedures

Oracle Automotive Trading Partner Toolkit provides a mechanism and tools to develop, add and maintain trading partner specific customizations separate from the base products. It allows you to rapidly capture, develop and execute a solution to meet trading partner or industry requirements.

Oracle developers register customizable program units in a central repository. Layer developers then provide trading partner specific extensions by writing code with a different processing behavior for the program units published in the repository. Next, these new program units can be gathered into sets, called layers, and seamlessly merged with base code for Oracle Automotive so that trading partner customizations are automatically recognized and instituted by the software. Additionally, Oracle Automotive Trading Partner Toolkit ensures that these layers are maintained independently from the underlying program units, and not overwritten by future bug fixes or patches as part of the Oracle base product maintenance.

Prerequisites

Before proceeding, you must set the values for the following profile options to ensure accurate processing:

- VEA: Layer Provider
- VEA: Customer

See the Oracle System Administrator's User's Guide for instructions on how to do this.

To proceed through the customer procedures listed in this chapter, use the following values:

Profile Option	Value
VEA:Layer Provider	CUST
VEA:Customer	<customer name agreed upon between you and layer developers>

Apply patches

You must first apply any patches shipped from Oracle or any trading partner layer developers. When inspecting the log files, look for errors and warnings which may be related to the Trading Partner Toolkit, and follow the instructions outlined in the error or warning messages.

Code Conversion Values

Code conversion provides mapping between a value specified in the branch criteria and the value as it exists in your ERP database. For example, the branch criteria might be

Trading Partner Group = Modern Truck.

However, in your ERP database, the trading partner group “Modern Truck” is identified as the value “MT”, but this must be mapped using the Layer Manage utility. This utility, run as part of ADPATCH, inserts these code conversion values and includes messages in the log file. The following is an example of one such message in the log file:

> “Please specify the internal value using ECE code conversion form for external value Modern Truck under code conversion category VEA_TP_GROUP_CODE with key1 as LP1.”

Develop Supplier-Specific Customizations

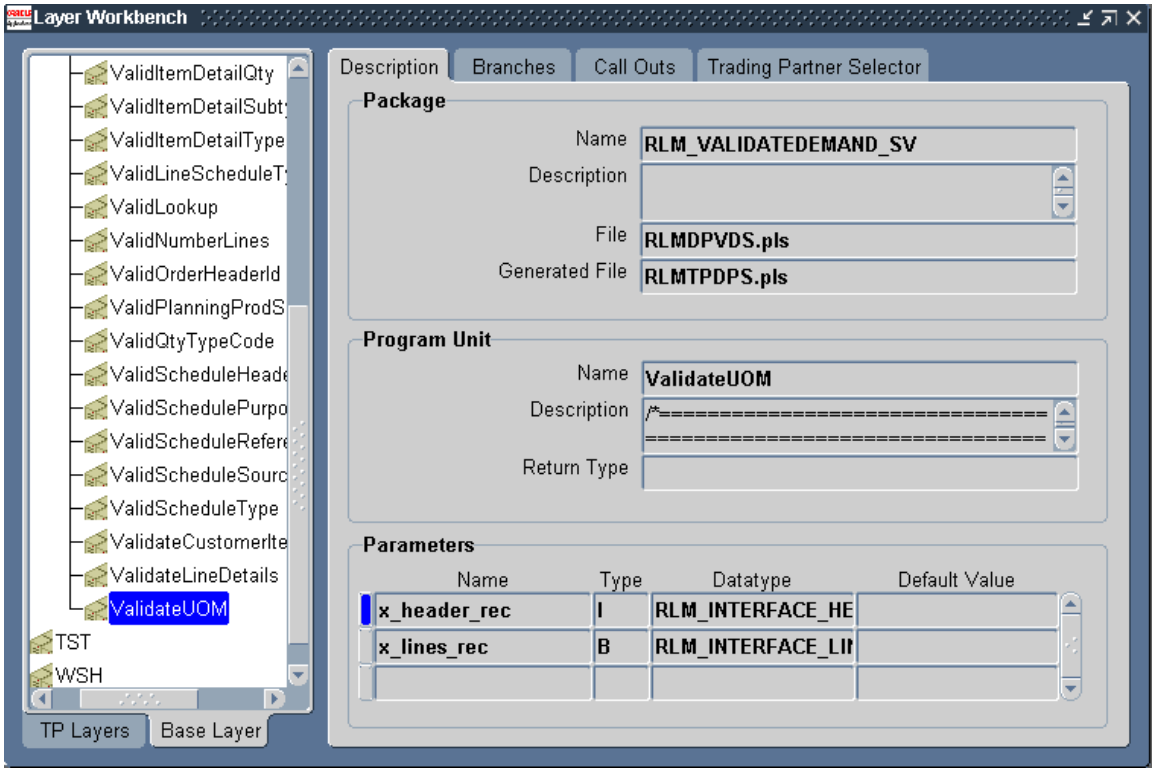
Complete the following steps to create Call Outs to allow supplier-specific customizations. All customizations specific to trading partners should be created as trading partner layers and will be shipped by layer providers. Be sure to use Call Outs for only those customizations specific to your site and not intended to distinguish trading partners from one another.

Step 1: Plan and Develop Supplier-Specific Code

Begin by analyzing your business requirements to identify the program units in the base layer that should include supplier-specific customizations. You can inspect the available base layer program units in the Layer Workbench.

◗◗ To examine base layer program units:

1. Through Automotive Manager>Trading Partner Toolkit>Layer Workbench, navigate to the Layer Workbench form.



2. Select the Base Layer tab on the left side and the Description tab on the right side of this form.
3. Expand the items in the Base Layer tree to examine the program units.
4. Select a program unit or package on the left side to view the name, description and further details on the right side of the window.
5. Take note of the program units that need supplier-specific code.

Note: Base Layer program units may be associated with, at most, two Call Out program units: one Call Out before and one Call Out after the base layer program unit.

For each base layer program unit you have identified for supplier-specific code changes, you must develop appropriate Call Out Before and Call Out After program units.

Note: Call Out program units should have no more IN parameters without a default value than those of its base layer counterpart.

Create your own PL/SQL packages to hold these Call Out program units. For each of the packages you create, ensure that the following naming prefixes for the package and its files are used:

- the package name should begin with the layer provider code and underscore character. The layer provider code of a customer is “CUST”. For example, CUST_ValidateUOM_For_Site15.
- the package file name should begin with “CUST”. For example, a valid package specification file would be CUSTDPS.pls, and a valid body file would be CUSTDPB.pls.

See Also

Layer Workbench

Step 2: Import Call Out Code

Next, you must import the Call Out code into Trading Partner Toolkit repository. The Call Out program units can then be linked to the appropriate base layer program unit in the Layer Workbench.

◗◗ To import the Call Out code into the repository:

1. Set up the oracle applications environment. Be sure that the environment variable TWO_TASK points to the correct database instance.
2. Run the import utility with the following command line:

   java oracle.apps.vea.main GWYUID=<value of environment variable GWYUID> TWO_TASK=<value of environment variable TWO_TASK>

FNDNAM=<value of environment variable FNDNAM> USERID=<Userid> FUNCTION=IMPORTALL APPLICATION=<App> FILE=<File Name>

where:

- USERID is the username and password, e.g. GLOBAL/GLOBAL93.
- APPLICATION is the application shortname, i.e. either RLM or WSH.
- FILE is the PL/SQL specification file name ending with S.pls. The file should be present in the current directory from which this utility is run.
- In this example, run the import utility on this file:

  java oracle.apps.vea.main GWYUID=$GWYUID TWO_TASK=$TWO_TASK FNDNAM=$FNDNAM USERID=GLOBAL/GLOBAL93 FUNCTION=IMPORTALL APPLICATION=RLM FILE=CUSTDPS.pls

3. Verify that no errors are displayed by the import utility.

See Also

Layer Workbench

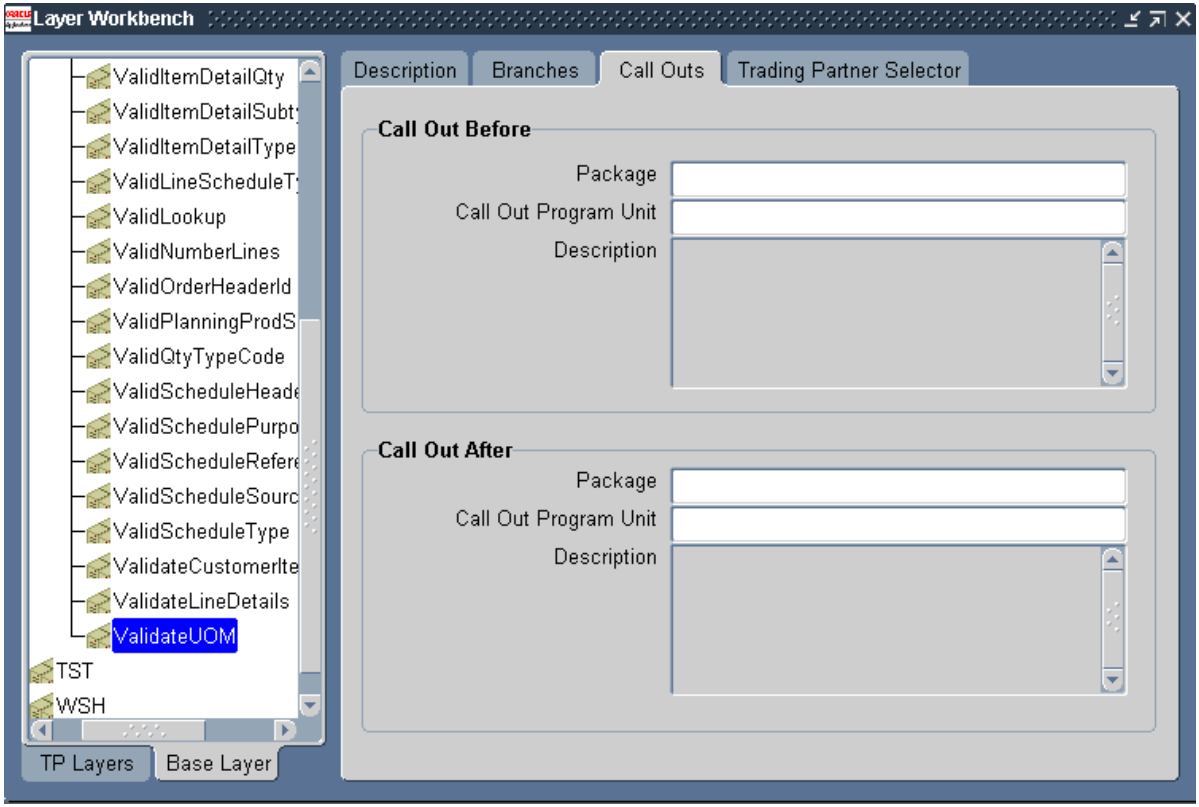
Import Utility

Step 3: Build Call Outs

After importing the Call Out code into the repository, you must create the Call Out to link the program unit to the appropriate base layer program unit in the Layer Workbench.

◗◗ To create a Call Out:

1. Through Automotive Manager>Trading Partner Toolkit>Layer Workbench, navigate to the Layer Workbench form.



2. Select the Base Layer tab on the left of this window.
3. Expand the items in the Base Layer tree to find the appropriate Base Layer program unit.
4. Select the program unit to list any associated Call Outs on the right side of the window.
5. Select the Call Outs tab on the right. If there are Call Outs attached to the current Base Layer program unit, you will see these listed here.

Note: Base layer program units may be associated with, at most, two Call Out program units: one Call Out before and one Call Out after the base layer program unit.

6. From the list of values in the appropriate region of the Call Outs tab, select the package name and Call Out program unit.
7. Save your work.

See Also

Layer Workbench

Step 4: Generate Call Out Code

After you have created the necessary Call Outs, follow the steps below to run the Generate utility to create the generated package files. For more information, see: Generate Utility on page 4-5.

To generate the Call Out code from the repository:

1. Set up the oracle applications environment. Verify that environment variable TWO_TASK points to the correct database instance.
2. Run the export utility with the following command line

   java oracle.apps.vea.main GWYUID=<value of environment variable GWYUID> TWO_TASK=<value of environment variable TWO_TASK> FNDNAM=<value of environment variable FNDNAM> USERID=<Userid> FUNCTION=GENERATE APPLICATION=<App> FILE=<File Name>

   where:

   - USERID is the username and password e.g. GLOBAL/GLOBAL93.
   - APPLICATION is the application shortname, i.e. either RLM or WSH.
   - FILE is the PL/SQL specification file name of a TP package ending with S.pls. For RLM, this will be the file name of the common TP package “RLMTPDPS.pls”. The specification and the body files for the specified TP package are created in the current directory.
   - In this example, the export command would be:

```
java oracle.apps.vea.main  GWYUID=$GWYUID TWO_TASK=$TWO_
TASK FNDNAM=$FNDNAM  USERID=GLOBAL/GLOBAL93
FUNCTION=GENERATE APPLICATION=RLM FILE=RLMTPDPS.pls
```

3. Verify that the specification and body files are created in the current directory.

See Also

Layer Workbench

Generate Utility

Step 5: Test the Call Outs

Compile the generated packages and any additional Call Out packages you have created in your testing database and verify that the desired functionality is achieved with the Call Outs.

> **Note:** For any generated package, you need to compile only package body; you should not compile package specification.

Activate a Trading Partner Layer

The Activate Trading Partner Layers window allows you to select a Trading Partner Layer and Layer Provider Code and display all corresponding program units based on this combination. You have the ability to activate and deactivate either an entire layer or individual program units.

Activate/De-activate All Layers or Callouts

You can set the **VEA: Customization Level** profile option to activate or deactivate all layers or, additionally, all layers and callouts. This profile option can have the following values:

Value	Description
Base Layer Only	All layers and callouts are inactive; generic functionality applies.
Base Layer and Trading Partner Layers Only	Trading Partner Toolkit checks the status of each layer as set in the Activate Trading Partner Layers window. All callouts are inactive.

Value	Description
Base Layer, Trading Partner Layers and Callouts	Trading Partner Toolkit checks the status of each layer as set in the Activate Trading Partner Layers window. All callouts are active.

To activate a specific trading partner layer:

1. Through Automotive Manager>Trading Partner Toolkit>Runtime on the menu, navigate to the Activate Trading Partner Layers window.
2. Select the Layer Name from the list of values.
3. Click Activate to activate the entire layer, or

   Click Activate for specific program units to activate only particular program units.
4. Save your work.

See Also

Layer Workbench

Activate / Deactivate Layers

Trading Partner Specific Attributes

Trading Partner Specific Translator Maps

The EDI translator should use the trading partner specific translator maps to handle any additional attributes.

4

Forms and Utilities

This chapter describes the Trading Partner Architecture Forms and Utilities included with Oracle Automotive and how to use them.

Layer Workbench

The Layer Workbench allows you to create trading partner layers and branches within these layers. The workbench consists of two regions: on the left, a hierarchical tree for viewing layers (including both trading partner and base layers), and on the right, a series of tabs describing each layer and branch in further detail.

Tabbed Regions - Left

- TP Layers--displays the program units which perform trading partner specific processing or validations.

  When you select this tab, the right side of the window displays corresponding information relating to the base layer program units, including package, base layer program unit and description, and branch criteria, including customer

number, trading partner group, ship to, intermediate ship to, and bill to addresses.

- Base Layer--displays the program units shipped by Oracle as part of the base release. When you select this tab, the right side of the window displays tabs corresponding to Branches, Call Outs, Trading Partner Selector and Generated Code.

Tabbed Regions - Right

- Branch--displays branch information relating to the trading partner layer, including package name, base layer program unit and description of the published program unit, as well as branch criteria, including customer number, trading partner group, shipping address, intermediate shipping address, and billing address.

The following tabs are only visible when the Base Layer tab is selected on the left of the window.

- Description--displays descriptive information for the package, including name, description, specification file and generated file, and for the program unit, including name, description, and return type. Also displays parameter information including name, type, data type and default value.
- Branches--displays corresponding branch information, including sequence, layer provider, layer name, layer program unit, customer number, trading partner group, shipping address, intermediate shipping address and billing address.
- Call Outs--displays call out information including package, call out program unit and description for call outs before and after the base layer program unit.

> **Note:** You can have a maximum of two Call Outs, one before and one after a Base Layer program unit.

- Trading Partner Selector--displays information corresponding to the trading partner selector and parameter mappings, including package, trading partner selector name, description, and the trading partner selector to base layer program unit parameter mapping.

Note: There can be only one Trading Partner Selector for each Base Layer program unit.

Trading Partner Layers window

The Trading Partner Layers window can be invoked through a pop-up menu on the right mouse button. This window includes layer name and description for editing or creating new layers.

Create Layer Licenses

The Create Layer Licenses window allows you to select a trading partner layer developed by the current layer developer and display all the customers licensed to that layer. This window also allows you to modify an existing license or create a new license.

At the top of the window is information about the layer, including Layer Name, Layer Provider Code and a Description of the layer. Below this is a list of customers licensed to the user-selected Trading Partner Layer Name and Layer Provider Code. They are listed by Customer Name and Description.

Note: To use this form, you must have created trading partner layers using the Layers Workbench.

For a description of how to create a trading partner layer license, see Step 10: Create Trading Partner Layer Licenses on page 2-17.

Activate Trading Partner Layers

The Activate Trading Partner Layers window allows you to select a trading partner layer and layer provider code and display all corresponding program units based on this combination. You have the ability to activate and deactivate either an entire layer or individual program units.

This window displays the trading partner layer, with layer provider code, and the particular program units associated with each layer. All fields are read-only except for the Activate flags that apply separately to the entire layer and each program unit. Each program unit is described in the bottom half of the window by application short name, package label, program unit label, and program unit name.

For a description of how to activate a trading partner layer, see Activate a Trading Partner Layer on page 3-9.

See Also

Create a Trading Partner Layer

Associate Trading Partner Layers with Base Layers

Create Trading Partner Layer Licenses

Develop Site-Specific Customizations

Overview of TP Utilities

Oracle Automotive Trading Partner Toolkit provides a mechanism to add and maintain trading partner specific customizations separate from the base products. Three separate utilities are included:

- Import

  The import utility can be used by layer developers to register code with the Trading Partner Toolkit repository. This is necessary for registering customizations included in trading partner layers.

- Generate

  The generate utility can be used by layer developers and customers to generate TP packages from the repository. This can be useful for extracting code, e.g. TP code or Call Outs, to generate package files for use at your site.

- Layer Manage

  The layer manage utility can be used by layer developers and customers to merge different layers with base product and existing layers.

Import Utility

The import utility registers program units (public, TP, TPS and custom) with the repository. Oracle developers insert TP tags in the code to indicate customizable or TPS program units and to associate each customizable program unit with a TP program unit. Layer developers can insert TP tags into their code to indicate TPS program units and to associate custom program units with specific trading partner layers. Once the source code is tagged, layer developers can run the import utility to parse the tagged code and register the program units with the repository.

Features

- Parses client-side libraries, i.e. *.pld files, and server-side packages, i.e. *.pls files

Command Line Syntax

java oracle.apps.vea.main GWYUID=<value of environment variable GWYUID> TWO_TASK=<value of environment variable TWO_TASK> FNDNAM=<value of environment variable FNDNAM> USERID=<Userid> FUNCTION=IMPORT APPLICATION=<App> [PATH=<Path>][FILE=<File Name>]

where:

- USERID is the username and password, e.g. GLOBAL/GLOBAL93.
- APPLICATION is the application shortname, i.e. either RLM or WSH.
- PATH is the location of PL/SQL specification/library files. This defaults to the current directory. Be sure to include a trailing "/" for this value, e.g. /myhome/tpafiles/
- FILE is the PL/SQL specification file name ending with S.pls. The file should be present in the current directory from which this utility is run.

> **Note:** Users must ensure the correct environment settings before running import utility. Please refer to Oracle Applications Installation manual for further details on how to set up environment.

Generate Utility

The Trading Partner Toolkit repository stores information about customizable program units and customizations(layers) developed by layer developers and customers. The generate utility reads this information and generates the code for TP packages.

Features

- Generates client-side libraries, i.e. *.pld files, and server-side packages, i.e. *.pls files
- Generates code for conditional execution of layer code
- Generates code only for layers licensed to the customer

- Generates code to perform code conversion for branching, i.e. customization, condition values
- Generates version information for use by ADPATCH
- Generates code to execute active layers only

Command Line Syntax

The Generate utility reads information from the repository about public program units and customizations, i.e. layers, created by layer developers or customers, then generates the code for TP packages.

The complete command line syntax is as follows:

java oracle.apps.vea.main GWYUID=<value of environment variable GWYUID> TWO_TASK=<value of environment variable TWO_TASK> FNDNAM=<value of environment variable FNDNAM> USERID=<Userid> FUNCTION=GENERATE APPLICATION=<App> [PATH=<Path>][FILE=<File Name>]

where:

- USERID is the username and password e.g. GLOBAL/GLOBAL93.
- APPLICATION is the application shortname, i.e. either RLM or WSH.
- PATH is the location of PL/SQL specification/library files. This defaults to the current directory. Be sure to include a trailing "/" for this value, e.g. /myhome/tpafiles/
- FILE is the PL/SQL specification file name of a TP package ending with S.pls. For RLM, this will be the file name of the common TP Package "RLMTPDPS.pls".

> **Note:** Verify that you are using the correct environment settings before running the generate utility. Please refer to Oracle Applications Installation manual for further details on how to set up your environment.

Layer Manage Utility

The layer manage utility provides a mechanism for you to deploy the repository and to merge TP metadata coming from various sources. The layer manage utility can be run in download and upload modes:

- Download: In download mode, the Layer Manage utility extracts TP metadata from the repository into an interface data file. This interface data file can be used to ship the repository to layer developers and customers. TP metadata must be shipped with any patches that include TP-enabled program units.
- Upload: In upload mode, the Layer Manage utility imports TP metadata from an interface data file and merges with existing data in repository. ADPATCH will run layer manage in upload mode before re-generating TP packages.

Features

- Flexible partitioning of TP metadata
  - Layer-wise partitioning across applications
  - Application-wise partitioning
  - Application-wise, layer-wise partitioning
  - Entire Trading Partner Toolkit repository
- Uses AOL generic loader (FNDLOAD)
- Uploads TP metadata only if Oracle Automotive is installed at the customer site. In this way, it enables both Oracle developers and layer developers to prepare the same patch for automotive and non-automotive customers.
- Automatically creates new e-Commerce gateway code conversion categories and values for branch criteria conditions.

> **Note:** Examine the log file after running the layer manage utility for any errors and notifications. If new EDI code conversion values were inserted, this log file will have notifications instructing you to map the branch condition value to appropriate internal value.

Use the following command line syntax for running the layer manage utility in download mode:

```
FNDLOAD logon 0 Y DOWNLOAD $VEA_TOP/admin/import/vea.lct
filename TPA [APP_SHORT_NAME=short_name] [LAYER_NAME=layer_
name]
```

where:

- logon is database userid/password[@connect_string]

- APP_SHORT_NAME is RLM OR WSH.
- filename is the name of TP metadata, including the full path, e.g. /myhome/tpa/RLMTPA.ldt

Note: Depending on your environment, the latest version of vea.lct may reside in $VEA_TOP/admin/import/vea.lct or in $VEA_TOP/patch/115/import/vea.lct. Please check with your system administrator and use the correct location in the syntax above.

Use the following command line syntax for running the layer manage utility in upload mode,

FNDLOAD logon 0 Y UPLOAD $VEA_TOP/admin/import/vea.lct filename

where:

- logon is database userid and password[@connect_string]
- filename is the name of TP metadata file, including the full path, e.g. /myhome/tpa/RLMTPA.ldt

Note: Depending on your environment, the latest version of vea.lct may reside in $VEA_TOP/admin/import/vea.lct or in $VEA_TOP/patch/115/import/vea.lct. Please check with your system administrator and use the correct location in the syntax above.

A

Oracle Automotive Trading Partner Toolkit User's Guide Code Restrictions

Restrictions on Imported Code

The following restrictions are applicable to any PL/SQL file imported into the Trading Partner Toolkit repository. This code can be imported by Oracle developers as part of the base layer, by layer developers as part of the trading partner layer, and by customers as part of site-specific Call Outs.

1. Packages and program units imported into the repository cannot be renamed or deleted.
2. Parameters of program units imported into the repository cannot be renamed or deleted.
3. All program units imported into the repository within a package must have unique names, i.e. PL/SQL program units cannot be overwritten by files of the same name.

B

Oracle Automotive Trading Partner Toolkit User's Guide Sample Driver Files for Layer Providers

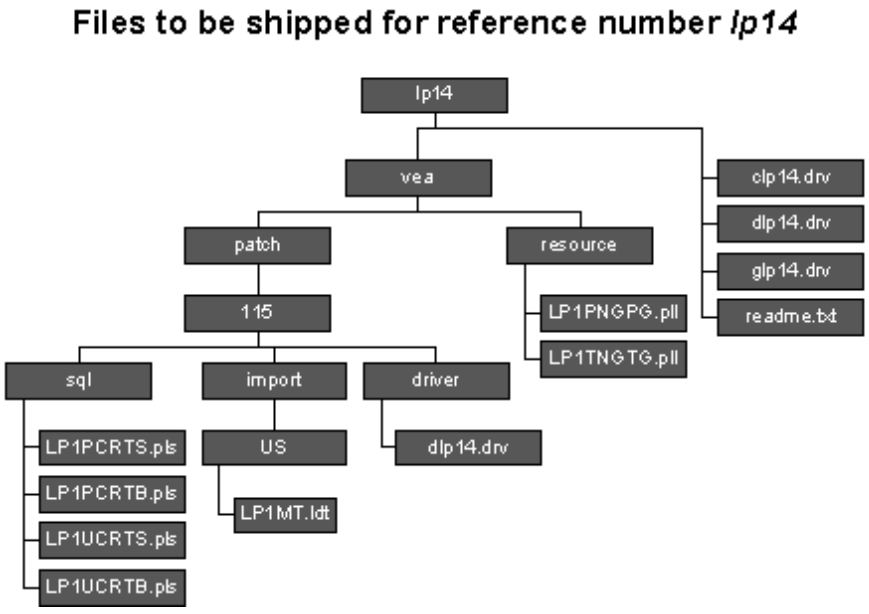
Driver Files

Driver files are prepared by layer developers after creating trading partner layers. A packaged trading partner layer includes these driver files.

The following objects are likely to be shipped by layer developers in the trading partner layer:

- packages - *.pls files
- repository metadata - *.ldt file
- form libraries - *.pll files
- flexfields - *.ldt files

The following sample driver files contain entries for each type of file a layer developer might need to package and ship to customers. As discussed earlier, each patch or packaged layer must be identified using a reference number. The following diagram illustrates the example used in the sample driver files and represents the stage area for a packaged trading partner layer with the reference number "lp14":



Copy Driver

This is a copy driver file, of the form c<reference number>.drv, responsible for copying files from patch download area to application top (APPL_TOP).

```
begin aru bug_lp14

  characterset us7ascii

  begin bug vea lp14
    begin actions
      copy      vea      patch/115/driver       dlp14.drv  115.1
      copy      vea      patch/115/sql          LP1PCRTS.pls 115.0
      copy      vea      patch/115/sql          LP1PCRTB.pls 115.0
      copy      vea      patch/115/sql          LP1UCRTS.pls 115.0
      copy      vea      patch/115/sql          LP1UCRTB.pls 115.0

      copy      vea      resource               LP1PNGPG.pll 115.0
      copy      vea      resource               LP1TNGTG.pll 115.0

  copy      vea      patch/115/import/US LP1MT.ldt 115.0
```

```
    exec   fnd     bin     FNLOAD bin &phase=last &ui_apps 0 Y UPLOAD
    @vea:patch/115/import/vea.lct  @vea:patch/115/import/US/LP1MT.ldt


    exec java oracle/apps/vea main.TPALoad.class java &phase=last+50 &un_apps
    &pw_apps &jdbc_protocol &jdbc_db_addr  &fullpath_rlm_patch/115/sql_
    RLMPTPAS.pls &env=GWYUID &env=TWO_TASK &env=FNDNAM


    exec java oracle/apps/vea main.TPALoad.class java &phase=last+50 &un_apps
    &pw_apps &jdbc_protocol &jdbc_db_addr  &fullpath_rlm_resource_RLMPNGTG.pll
    &env=GWYUID &env=TWO_TASK &env=FNDNAM


    end actions

  end bug vea lp14

end aru bug_lp14
```

Database Driver

This is a database driver file, of the form d<reference number>.drv, responsible for running SQL scripts and programs that update the database. Database driver file must begin with instructions to

- run layer manage utility in upload mode to upload layer metadata file
- generate trading partner packages

All other commands and instructions should come after the above.

```
begin aru lp14DB
  compatible release  11.5.0
  compatible parallel  no

  begin bug vea lp14

    begin prereqs
    end   prereqs

    begin actions
```

```
    #      # ----- CREATE PACKAGE SPECIFICATIONS (phase=pls) -------------------
    sql     vea      patch/115/sql      LP1PCRTS.pls none none none package
    &phase=pls
    sql     vea      patch/115/sql      LP1UCRTS.pls none none none package
    &phase=pls
    sql     rlm      patch/115/sql      RLMPTPAS.pls none none none package
    &phase=pls



    #      # ----- CREATE PACKAGE BODIES (phase=plb) --------------------------
    sql     vea      patch/115/sql      LP1PCRTB.pls none none none package
    &phase=plb
    sql     vea      patch/115/sql      LP1UCRTB.pls none none none package
    &phase=plb
    sql     rlm      patch/115/sql      RLMPTPAB.pls none none none package
    &phase=plb

    end actions

end bug vea lp14
end aru lp14DB
```

Generate Driver

This is a generate driver file, of the form g<reference number>.drv, responsible for generating PL/SQL library files and forms.

```
begin aru bug_lp14

  characterset us7ascii

  begin bug vea lp14
    begin prereqs
    end prereqs
    begin actions
      genfpll   vea       resource                LP1PNGPG.pll
      genfpll   vea       resource                LP1TNGTG.pll
      genfpll   rlm       resource                RLMPNGTG.pll
    end actions
  end bug vea  lp14

end aru bug_lp14
```

Glossary

B

Base Layer

The generic code independent of a Trading Partner. It consists of PL/SQL program units published as customizable by Oracle Development teams. Trading Partner Layers can be built only on those program units designated by an Oracle Development team as published.

branch

A link between a Trading Partner Layer program unit and a Base Layer program unit.

C

call out

A site-specific customization independent of a Trading Partner.

customer model serial number

In the Automotive industry, this is the Vehicle Identification Number (VIN).

L

layer

Encapsulates the trading partner specific modifications to Oracle code. This is equivalent to a trading partner library. A layer consists of a set of PL/SQL program units that perform trading partner specific processing or validations. Layer

Providers create Trading Partner Layers by developing and importing trading partner specific code into cohesive layers which can be shipped as a single unit.

layer provider

An organization or entity that builds layers for Oracle Automotive Trading Partner Architecture.

P

Package level tags

Package level tags can appear anywhere after a “CREATE OR REPLACE” statement and before any uncommented package contents, including variables, program units, etc. For example,

--<TPA_LAYER=layer name>

indicates that the package belongs to the specified Trading Partner Layer.

Program Unit

Any packaged PL/SQL procedure or function.

Program Unit Level Tags

Program unit level tags must appear immediately after keyword 'IS'.

TPS Program Unit: --<TPA_TPS>

Public Program Unit

Those program units published as customizable by Oracle Development teams. Layers can be built only on those program units that are designated by an Oracle Development team as public. These may also be referred to as “published” or “customizable” program units.

T

tare weight

The weight of an item, excluding packaging or included items.

TPA metadata file

Contains information extracted from the TPA repository about TPA enabled program units and layers built on top of them. This file is used to ship the TPA registry, or repository, and merge layers at the customer site. This file must be shipped with any patch that contains TPA enabled program units.

TPA package

The package containing TPA program units. This package is always generated from the TPA repository.

TPA program unit

The mirror program unit for a public program unit. For every public program unit, Oracle developers will designate a TPA program unit. TPA program units are generated by the architecture to insulate generic code from custom code. All calls to customizable generic code and custom code are made through the TPA program unit.

TPA repository

The registry which stores data required for the functioning of the Trading Partner Architecture. It includes information about public program units, TPA program units, TPS program units and complete definition of the layers including the Oracle Base Layer.

TPA tag

One-line hyphen comments which appear at the beginning of a new line and provide information about customizable program units within Oracle code. The syntax for a TPA tag is:

```
--<tag name=tag value>
```

For example, a label is specified as follows,

```
--<TPA_LABEL=label>
```

trading partner

Any company that sends and receives documents via EDI.

Trading Partner Architecture (TPA)

The framework that supports PL/SQL based layer development and deployment.

trading partner flexfield

Descriptive flexfields reserved on several base tables for capturing additional attributes applicable to specific trading partners. They are provided for most of the base tables in Oracle Release Management, Shipping and Order Management.

trading partner layer

The trading partner specific code created to replace Base Layer code. The layer consists of a set of PL/SQL program units that perform trading partner specific processing or validations in place of the generic code provided by Oracle Development.

Layer Providers develop this code and populate the Trading Partner Layers by importing the trading partner specific code into the TPA repository. In this way, Layer Providers can develop Trading Partner Layers composed of trading partner specific code for various trading partners.

Trading Partner Selector (TPS)

A program unit which accepts context information for the business transaction and derives trading partner entities being processed in the current transaction instance.

All TPS Program units must have the following five output (OUT/IN OUT) arguments:

Name	Argument
Trading Partner Group Code	x_tp_group_code
Customer Number	x_customer_number
Ship To EDI Location Code	x_ship_to_ece_locn_code
Intermediate Ship To EDI Location Code	x_inter_ship_to_ece_locn_code
Bill To EDI Location Code	x_bill_to_ece_locn_code

V

vehicle

An exact instance of a vehicle type (for example, truck123). This information is sent to the customer through the Advance Ship Notice.

vehicle type

The outermost container, such as a truck or railcar.

Index

R

S

T

U

V

W