Information Builders

iWay

iWay Application Adapter for Oracle E-Business Suite User's Guide

Version 7.0.x and Higher

August 16, 2018

Contents

Preface

This document describes the iWay Application Adapter for Oracle E-Business Suite and how to use it for developing services and events for integration with other EIS applications.

Note: This Release 7.0.x content is currently being updated to support iWay Release 8.0.x software. In the meantime, it can serve as a reference for your use of iWay Release 8. If you have any questions, please contact *Customer_Success@ibi.com*.

How This Manual Is Organized

This manual includes the following chapters:

	Chapter/Appendix	Contents
1	Introducing the iWay Application Adapter for Oracle E-Business Suite	Introduces the iWay Application Adapter for Oracle E-Business Suite and describes its key features.
2	Application Adapter for Oracle E-Business Suite Supported Platforms Matrix	Specifies version, platform, and database support information for iWay Application Adapter for Oracle E-Business Suite User's Guide.
3	Application Adapter for Oracle E-Business Suite Quick Start Guide	Provides a quick start guide for the iWay Application Adapter for Oracle E-Business Suite.
4	Installing the iWay Oracle API Wrapper	Describes how to install/upgrade the iWay Oracle API wrapper, which is used to interact with Oracle APIs.
5	Configuring and Managing Connections to Oracle E-Business Suite	Describes how to configure and manage connections to Oracle E-Business Suite using iWay Explorer.
6	Integrating With Oracle Interface Tables and Concurrent Programs	Describes how to browse metadata, generate XML schemas, and create iWay Business Services for Oracle interface tables and concurrent programs.
7	Integrating With Oracle APIs	Describes how to browse metadata, generate XML schemas, and create iWay Business Services for Oracle APIs.

	Chapter/Appendix	Contents
8	Integrating With Oracle Base Tables	Describes how to browse metadata, generate XML schemas, and create iWay Business Services for Oracle base tables.
9	Creating Prepared Statements and Batch Statements	Describes how to create prepared statements and batch statements using iWay Explorer.
10	Listening for Oracle E-Business Suite Events	Describes how to listen for Oracle E-Business Suite events. Several listening techniques are available, enabling you to choose the technique that best suits your requirements.
11	Oracle E-Business Suite Troubleshooting and Error Messages	Describes limitations and workarounds when connecting to Oracle E-Business Suite.
A	Configuring the Application Adapter for Oracle E-Business Suite in an iWay Environment	Describes how to configure the adapter in the Service Manager console.
B	Configuring the Application Adapter for Oracle E-Business Suite in iWay Integration Tools Designer	Describes how to configure the adapter in iWay Integration Tools (iIT) Designer.
C	Supported Interface Tables for Oracle Release Apps 11i	Describes the agents that constitute Oracle functional components.
D	Sample WSDL File	Provides an example of a WSDL file generated from a web service using iWay Explorer.

Documentation Conventions

The following table describes the documentation conventions that are used in this manual.

Convention	Description
THIS TYPEFACE or this typeface	Denotes syntax that you must enter exactly as shown.

Convention	Description
this typeface	Represents a placeholder (or variable), a cross-reference, or an important term. It may also indicate a button, menu item, or dialog box option that you can click or select.
underscore	Indicates a default setting.
Key + Key	Indicates keys that you must press simultaneously.
{ }	Indicates two or three choices. Type one of them, not the braces.
|	Separates mutually exclusive choices in syntax. Type one of them, not the symbol.
...	Indicates that you can enter a parameter multiple times. Type only the parameter, not the ellipsis (...).
. . .	Indicates that there are (or could be) intervening or additional commands.

Related Publications

Visit our Technical Documentation Library at *http://documentation.informationbuilders.com*. You can also contact the Publications Order Department at (800) 969-4636.

Customer Support

Do you have any questions about this product?

Join the Focal Point community. Focal Point is our online developer center and more than a message board. It is an interactive network of more than 3,000 developers from almost every profession and industry, collaborating on solutions and sharing tips and techniques. Access Focal Point at *http://forums.informationbuilders.com/eve/forums*.

You can also access support services electronically, 24 hours a day, with InfoResponse Online. InfoResponse Online is accessible through our website, *http://www.informationbuilders.com*. It connects you to the tracking system and known-problem database at the Information Builders support center. Registered users can open, update, and view the status of cases in the tracking system and read descriptions of reported software issues. New users can register immediately for this service. The technical support section of *http://www.informationbuilders.com* also provides usage techniques, diagnostic tips, and answers to frequently asked questions.

Call Information Builders Customer Support Services (CSS) at (800) 736-6130 or (212) 736-6130. Customer Support Consultants are available Monday through Friday between 8:00 a.m. and 8:00 p.m. EST to address all your questions. Information Builders consultants can also give you general guidance regarding product capabilities and documentation. Please be ready to provide your six-digit site code number (*xxxx.xx*) when you call.

To learn about the full range of available support services, ask your Information Builders representative about InfoResponse Online, or call (800) 969-INFO.

Help Us to Serve You Better

To help our consultants answer your questions effectively, be prepared to provide specifications and sample files and to answer questions about errors and problems.

The following tables list the environment information our consultants require.

Platform	
Operating System	
OS Version	
JVM Vendor	
JVM Version	

The following table lists the deployment information our consultants require.

Adapter Deployment	For example, JCA, Business Services Provider, iWay Service Manager
Container	For example, WebSphere

Version	
Enterprise Information System (EIS) - if any	
EIS Release Level	
EIS Service Pack	
EIS Platform	

The following table lists iWay-related information needed by our consultants.

iWay Adapter	
iWay Release Level	
iWay Patch	

The following table lists additional questions to help us serve you better.

Request/Question	Error/Problem Details or Information
Did the problem arise through a service or event?	
Provide usage scenarios or summarize the application that produces the problem.	
When did the problem start?	
Can you reproduce this problem consistently?	
Describe the problem.	
Describe the steps to reproduce the problem.	
Specify the error message(s).	

Request/Question	Error/Problem Details or Information
Any change in the application environment: software configuration, EIS/database configuration, application, and so forth?	
Under what circumstance does the problem *not* occur?	

The following is a list of error/problem files that might be applicable.

- ❑ Input documents (XML instance, XML schema, non-XML documents)
- ❑ Transformation files
- ❑ Error screen shots
- ❑ Error output files
- ❑ Trace files
- ❑ Service Manager package to reproduce problem
- ❑ Custom functions and agents in use
- ❑ Diagnostic Zip
- ❑ Transaction log

For information on tracing, see the *iWay Service Manager User's Guide*.

User Feedback

In an effort to produce effective documentation, the Technical Content Management staff welcomes your opinions regarding this document. Please use the Reader Comments form at the end of this document to communicate your feedback to us or to suggest changes that will support improvements to our documentation. You can also contact us through our website, *http://documentation.informationbuilders.com/connections.asp*.

Thank you, in advance, for your comments.

Information Builders Consulting and Training

Interested in training? Information Builders Education Department offers a wide variety of training courses for this and other Information Builders products.

For information on course descriptions, locations, and dates, or to register for classes, visit our website (*http://education.informationbuilders.com*) or call (800) 969-INFO to speak to an Education Representative.

Chapter 1

Introducing the iWay Application Adapter for Oracle E-Business Suite

This section introduces the iWay Application Adapter for Oracle E-Business Suite and describes its key features.

The iWay Application Adapter for Oracle E-Business Suite enables you to reuse your existing Oracle E-Business Suite procedures and applications with other applications, a key to building a successful e-business or integrated enterprise.

In this chapter:

Features of the iWay Application Adapter for Oracle E-Business Suite

The iWay Application Adapter for Oracle E-Business Suite provides simple open standard access to Oracle E-Business Suite through Oracle E-Business Suite open interface tables and custom interface tables, stored procedures under a package, and direct interaction with base tables and views. No recoding or modifications of the Oracle E-Business Suite system are required.

In addition, the iWay Application Adapter for Oracle E-Business Suite enables you to fully integrate an Oracle E-Business Suite system with other enterprise resources, such as a Database Management System (DBMS) that has a JDBC™ 2.0–compliant driver, email system, HTTP protocol–based server, or FTP server.

From the adapter you also can connect to a host of enterprise integration systems (EIS), including popular enterprise resource planning (ERP), supply chain management (SCM), and customer relationship management (CRM) applications.

The iWay Application Adapter for Oracle E-Business Suite comprises several execution agents that enable integration with other systems by functioning in one of two ways:

- **Requests for Oracle inbound processing.** The adapter sends requests to the Oracle E-Business Suite system through Oracle interface tables and custom interface tables, stored procedures under a package, base tables and views, and Oracle published APIs.
- **Listeners for Oracle outbound events.** The adapter listens for application-based table activity.

Key Features of the iWay Application Adapter for Oracle E-Business Suite

Key features of the iWay Application Adapter for Oracle E-Business Suite include:

- Requests that communicate with Oracle published interface tables and custom interface tables, stored procedures under a package, base tables and views, and Oracle published APIs.
- Asynchronous as well as synchronous processing.
- Bidirectional message/request processing.
- XML-based requests and responses.
- Back-end error propagation through the RDBMS Table Listener.
- Oracle business logic and functionality that ensures accuracy and maintainability.
- iWay Explorer, which enables you to browse Oracle E-Business Suite metadata and generate XML schemas for service requests.

Integration with Oracle E-Business Suite

Using iWay Application Adapter for Oracle E-Business Suite, you can now interact with Oracle E-Business Suite in three ways: using Oracle interface tables and custom interface tables, stored procedures under a package, or base tables and views. These integration methods are listed and briefly described in the following section.

- Integration Using **Oracle Interface Tables and Custom Interface Tables**

Interface tables are one mechanism Oracle E-Business Suite uses to expose its business processing. Through this mechanism, data is never modified directly in Oracle E-Business Suite base tables and views. The iWay Application Adapter for Oracle E-Business Suite enables you to import data into Oracle E-Business Suite using specific Oracle-supplied interface tables or custom interface tables. Instead of transforming input documents into standards-based XML structures, the request document is transformed to match the appropriate Oracle-specific interface table.

Oracle recommends that users do not interact directly with application tables. The open interface APIs serve as bridges between the Oracle application modules and external systems. The iWay Application Adapter for Oracle E-Business Suite utilizes these tables to interact with Oracle. They are categorized according to functional area. This is achieved as a two-step process through the adapter. First, a record is inserted into the interface table and then an Oracle-supplied concurrent program moves the data from interface to base tables and views, ensuring that all business logic and processing is handled through Oracle-authored components. In Oracle E-Business Suite, concurrent processing simultaneously executes programs running in the background with online operations to fully utilize your hardware capacity.

For more information on integration using Oracle interface tables or custom interface tables, see *Browsing Interface Table Metadata* on page 67.

- Integration Using **Oracle Stored Procedures Under a Package (PL/SQL APIs)**

  Oracle E-Business Suite supports PL/SQL blocks, package. A package is a PL/SQL construct that allows related objects to be stored together. The adapter is able to expose all the packages and interact with its stored procedure element. In the iWay Application Adapter for Oracle E-Business Suite, packages are categorized by the schema that owns the object. Using iWay Explorer, you can search for specific packages and specific stored procedures.

  Note: If you do not see a package under a schema, you may need to create a public synonym. For more information, see *How to Search for Packages* on page 88.

- Direct Interaction with **Oracle Base Tables and Views**

  You can now interact directly with Oracle E-Business Suite base tables, bypassing interface tables and concurrent programs. This integration method is most appropriate when you need to query data located in Oracle base tables and views.

  Important: The use of INSERT, DELETE, or UPDATE to directly modify data in Oracle base tables is not recommended, as it may jeopardize database referential integrity.

For more information on interacting with Oracle base tables and views, see *Browsing Interface Table Metadata* on page 67.

Using the Adapter with Oracle APIs

The iWay Application Adapter for Oracle E-Business Suite uses an iWay wrapper to interact with Oracle APIs with complex data types, such as table and record types. The wrapper, which must be manually installed, contains two stored procedures, get_api_definition and run_api. These stored procedures are internally executed as 'APPS' user. The adapter interacts with the wrapper at design-time through the get_api_definition stored procedure, and at run-time through the run_api stored procedures. The adapter uses an Oracle thin driver, such as ojdbc14.jar, to interact with the Oracle database. For installation instructions, see *Installing the iWay Oracle API Wrapper* on page 41.

Note: It is presumed that the following initialization parameters are required to run the standard APIs; org_id, application_id, responsibility_id, and user_id.

An additional parameter (iway_oraapps) is also included in the XML request schema to support APIs in non-Oracle application environments.

iWay Oracle Wrapper Architecture

The iWay Application Adapter for Oracle E-Business Suite interacts with Oracle APIs through an iWay Oracle wrapper procedure called iway_ora_api_wrapper which is installed under the 'iway' user. The wrapper has two stored procedures- get_api_definition and run_api. These stored procedures are internally executed as 'APPS' user. The following topics describe how the adapter interacts with the wrapper at design time and run time.

Design Time

The adapter queries Oracle's sys.all_objects table to build the list of packages and stored procedures. When a user navigates to a stored procedure within a package and clicks the 'stored procedure call' node the adapter sends the name of the API in the form of <schema.package_name.stored procedure_name> to call the get_api_definition procedure. The get_api_definition procedure queries the data dictionary in Oracle and retrieves the required information and does the following:

1. Delete any old values and Insert the parameter information to the IWAY_API_PARAMETERS table.
2. Build the request_schema and response_schema values and send this information to the adapter. Also the status of the get_api_definition execution is also returned to the adapter in the wrapper_status variable.

The adapter in turn uses this information from the request_schema and response_schema variables to create the request xsd and response xsd schemas and eventually the WSDL.

Run Time

When a web service is invoked, the XML payload is taken by the adapter and the API name, parameters name, data types, and values are passed to the run_api script in the wrapper. The wrapper then executes the Oracle API and sends the output to the adapter. It also sends any Oracle error messages to the adapter. The adapter displays this output as its response.

Understanding the Functionality of the RUN_API Script

The run_api script receives an input file from the adapter to execute. It uses the api_parameters table to validate the names of the input parameters that are passed from the adapter and to determine the data type for those input parameters. If the metadata does not exist for the given API, it executes get_api_def to check the data dictionary and loads the data into the api_paremeters table.

The wrapper then checks if the dynamic declaration file exists, which is needed to hold the variables for the given API. If the dynamic declaration file does not exist, it is automatically created. The run_api script runs a query against the dba_objects table at the beginning of each execution, which is part of its dynamic declaration processing. The dynamic declaration files are generated in real time.

The dba_objects table is checked during each execution of the wrapper to see if a declaration file exists for the API with a creation date greater than or equal to the API creation date. If this is true, the run_api script goes to the next step. If this is false, the API generates the declaration file. Global variables are then created to store the input parameters. These global variables are represented using the following format:

```
iway2_xxx.parameter_name
```

For example:

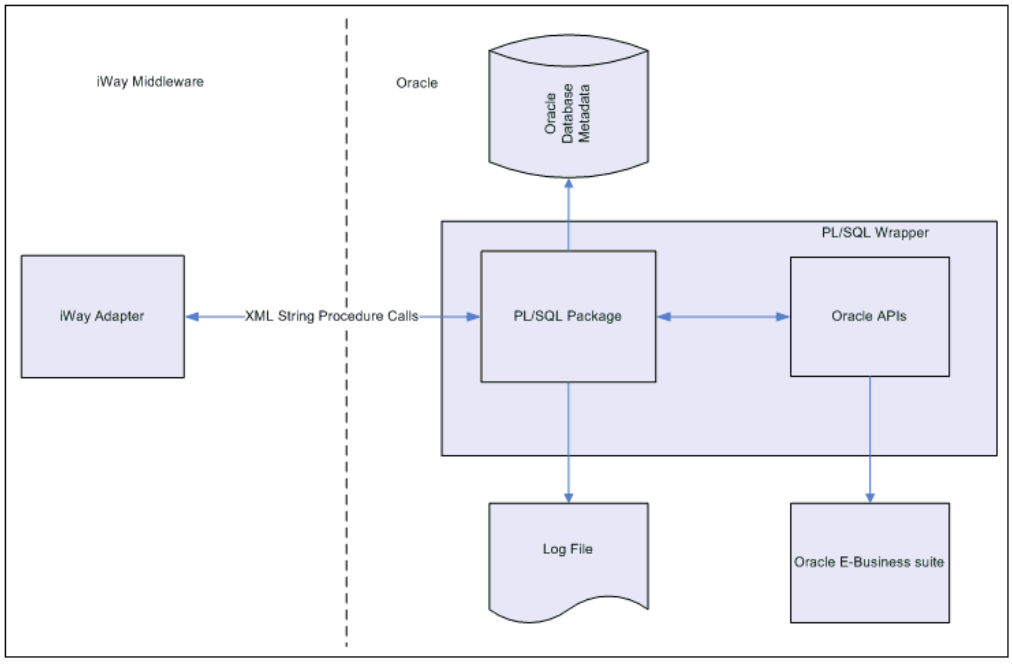
```
P_API_VERSION => IWAY2_159656_Process_Items.P_API_VERSION
```

Next, the input parameter processing routine runs to move the input parameters into the global variables. This includes interpreting the delimiters to properly support tables and record abstract data types. Variables are also prepared in the correct datatype. After all of the input parameters have been processed, the API call string is generated. This is the actual statement that is sent to the API (and is documented each time the log file is executed).

If you review the call string in the wrapper log file, you will notice that it is a simple call string that uses the same syntax you would use if you were calling the API from sql*plus in a pl/sql code snippet. This is the only direct connection between the wrapper and the API. The wrapper then waits for the API to complete and then parses the output from the API into a syntax which can be passed back to the adapter.

iWay Oracle Component Architecture

The adapter drives the necessary transactional data for the Oracle APIs. The wrapper acts as an intermediary layer to pass and receive transactional data from and to the APIs and communicate that back to the adapter.



The following topics describe the GET_API_DEFINITION and RUN_API procedures in more detail.

GET_API_DEFINITION

GET_API_DEFINITION Parameters:

Parameter Name	Mode	Data Type	Description
API_NAME	IN	VARCHAR2	The name of the API to be called by the wrapper.
REQUEST_SCHEMA	OUT	VARCHAR2	API Input parameter and data type string.
RESPONSE_SCHEMA	OUT	VARCHAR2	API Output parameter and data type string.
DEBUG_LEVEL	IN	NUMBER	0 = Debug off. 1 = Debug on.
WRAPPER_STATUS	OUT	VARCHAR2	Wrapper status.

REQUEST SCHEMA:

The following initialization parameters are required to run the standard APIs: org_id, application_id, responsibility_id, and user_id. These will be passed as part of the request schema, although they are not generated from the data dictionary. Each request_schema generated by the get_api_definition will start with the four required initialization parameters as follows:

```
REQUEST_SCHEMA = 'ORG_ID|||NUMBER|||USER_ID|||NUMBER|||
APPLICATION_ID|||NUMBER|||RESPONSIBILITY_ID|||NUMBER|||..........'
```

IN OUT PARAMETERS:

Parameters with a mode of IN/OUT will be passed in both the REQUEST and RESPONSE schema. There will be no other designation for IN/OUT parameters.

CUSTOM TABLE FOR IN/OUT PARAMETERS:

A custom table, iway_api_parameters, will be loaded with the IN/OUT parameters for a given API when that API is called in the get_api_definition procedure. The procedure will remove any existing records for the given API and then load the table with the appropriate IN/OUT parameters. This will be used by the run_api procedure as described in the following section.

DELIMITERS:

The field delimiter is |||, the record delimiter (which is used for pl/sq advanced record types) is ###, and the table type delimiter is ***.

PARAMETER FORMAT:

1. Table type parameter name.
2. Table type delimiter.
3. Field name or null value.
4. Field delimiter.
5. Datatype.

Example metadata for table based on a pl/sql record:

In this example, the table type parameter name = 'P_LINES_TABLE' which is based on a pl/sql record with three fields, 'link_to_line_index', 'rev_exist', 'unearn_exist'.

```
P_LINES_TABLE***LINK_TO_LINE_INDEX|||BINARY_INTEGER|||
P_LINES_TABLE***REV_EXIST|||VARCHAR2|||
P_LINES_TABLE***UNEARN_EXIST|||VARCHAR2
```

Example of table based on varchar2:

In this example, when tables are comprised of a scalar datatype, the table type parameter name = 'P_TBL_VCHR' and there are no field names.

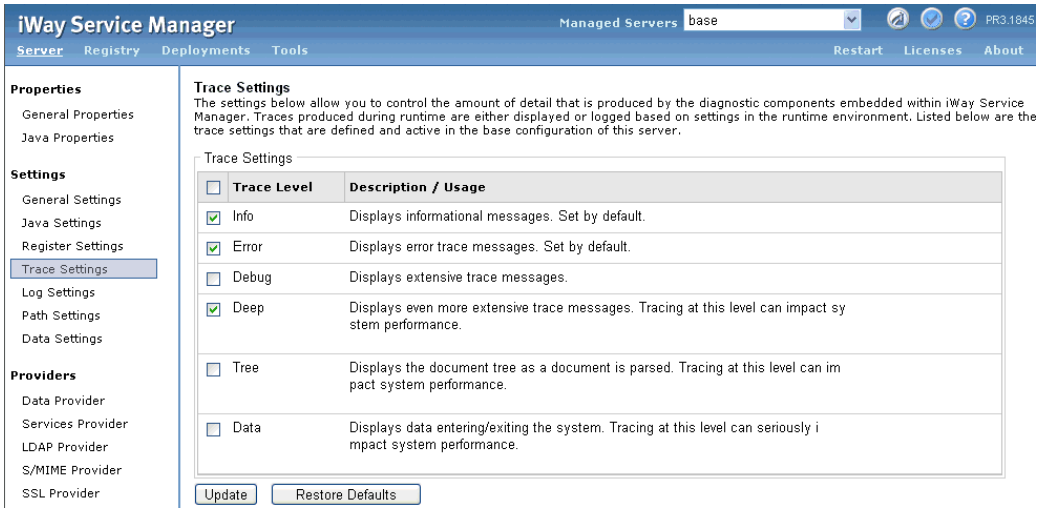
```
P_TBL_VCHR***|||VARCHAR2
```

Since there is no field name between the table delimiter and the field delimiter, it is a table based on a scalar datatype of varchar2.

DEBUG LEVEL:

The debug_level parameter is passed with a value of 1, so that the wrapper will write messages to the log file, IWAY_WRAPPER_LOG.txt. You can change this setting in the iway_ora_api_wrapper.pks file and recompile the wrapper.

For more detailed debugging, you must logon to the iWay Service Manager Administration Console and modify the trace settings by selecting the *Deep* check box, as shown in the following image.



Click *Update* when you are finished.

For more information on how to modify trace settings using the iWay Service Manager Administration Console, see the *iWay Service Manager User's Guide*.

WRAPPER STATUS:

The wrapper status parameter returns wrapper specific success or failure messages. For example, if you pass an incorrect API name to the get_api_definition, the wrapper status will be as follows:

```
WRAPPER_RETURN_STATUS|||ERROR|||WRAPPER_RETURN_CODE|||
INVALID_API_NAME|||WRAPPER_RETURN_MESSAGE|||
```

Invalid API name supplied. Cannot obtain Parameter Definitions from Oracle.

Parsed, it looks like this:

WRAPPER_RETURN_STATUS	ERROR
WRAPPER_RETURN_CODE	INVALID_API_NAME
WRAPPER_RETURN_MESSAGE	Invalid API name supplied. Cannot obtain Parameter Definitions from Oracle.

RUN_API

RUN_API Parameters:

Parameter Name	Mode	Data Type	Description
API_NAME	IN	VARCHAR2	The name of the API to be called on by the wrapper.
API_PARAMETER_VALUES	IN	VARCHAR2	String with the parameter_name, data type, and value based on the REQUEST_SCHEMA of the get_api_definition procedure.
API_OUTPUT_PARAMETERS	IN	VARCHAR2	The definition of the API output parameters. These parameters are needed as placeholders during the dynamic call to the APIs.
API_RETURN_STATUS	OUT	VARCHAR2	API output values, error codes and messages.
DEBUG_LEVEL	IN	NUMBER	0 = Debug off. 1 = Debug on.

API PARAMETER VALUES

The adapter sends back the four initialization parameters defined in the get_api_definition request_schema for a given API. These are required parameters. The parameters are passed as part of the normal request_schema in the get_api_definition procedure.

IN OUT PARAMETERS:

The run_api parameter will test parsed API parameters from the API_PARAMETER_VALUES wrapper input parameter against the iway_api_parameter table to determine if any parameters are of mode IN/OUT. Those API parameters that are IN/OUT will be assigned a variable, the value passed in the api_parameter_values parameter will be assigned to that variable, and the variable will be passed to the API as an IN/OUT parameter. Those API parameters that are IN parameters will be passed with the actual value provided.

The API_OUTPUT_PARAMETERS response_schema parameter is also tested to ensure that the IN/OUT parameters already added to the call string as part of the API_PARAMETER_VALUES parsing are excluded. This is to make sure that the IN/OUT parameter is not added twice to the dynamically built call string.

The RETURN_STATUS parameter strings together all of the OUT and IN/OUT parameters, and their associated values, passed as part of the API_OUTPUT_PARAMETERS parameter and returns them to the adapter.

DELIMITERS:

The field delimiter is |||, the record delimiter is ### (which is used for pl/sq advanced record types), and the table type delimiter is ***.

PARAMETER FORMATS:

For run time, the format will be based on table type parameters as follows:

When data is passed for table type parameters from the adapter to the wrapper in run_api, the adapter must pass the parameter name, the table row number, the table field name if the table is based on PL/SQL record, and the actual data value if it is based on scalar datatype.

1. Table type parameter name.
2. Table type delimiter.
3. Field name or null value.
4. Field delimiter
5. Data

Example of table based on a pl/sql record:

In this example, the table type parameter name = 'P_LINES_TABLE' which is based on a pl/sql record with three fields, 'link_to_line_index', 'rev_exist', 'unearn_exist'.

```
P_LINES_TABLE(1)***LINE_RECLINK_TO_LINE_INDEX|||103|||
P_LINES_TABLE(1)***REV_EXIST|||YES|||
P_LINES_TABLE(1)***UNEARN_EXIST|||NO|||
P_LINES_TABLE(2)***LINK_TO_LINE_INDEX|||2|||
P_LINES_TABLE(2)***REV_EXIST|||NO|||
P_LINES_TABLE(2)***UNEARN_EXIST|||YES|||
```

Example of table based on varchar2:

In this example, when tables are comprised of a scalar datatype, the table type parameter name = 'P_TBL_VCHR' and there are no field names.

```
P_TBL_VCHR(1)***|||Hi There - This is row 1|||
P_TBL_VCHR(2)***|||Hi There - This is row 2|||
P_TBL_VCHR(1)***|||And so on and so on...|||
```

API RETURN STATUS

The run_api procedure uses API_RETURN_STATUS to provide API and wrapper specific messages.

DEBUG LEVEL:

When the debug_level parameter is passed with a value of 1, the wrapper will write messages to the log file.

DATA TYPE CONVERSIONS:

Oracle API data type	Converted Adapter schema data type
BIT	Boolean
BOOLEAN	Boolean
TINYINT	Short
SMALLINT	Short
INTEGER	Int
BIGINT	Long
FLOAT	Double
REAL	Double
DOUBLE	Double
NUMERIC	Double
NUMBER	Double
DECIMAL	Int

All other oracle API data types will be converted and represented as 'string' in the schemas.

Additional Parameters Needed to Run an API

The wrapper generates additional parameters that are sent to the adapter when generating schemas. These are special parameters that are needed to run any Stored procedure.

Org_ID: The ID of the organization that the user needs to execute the API. For example, the ID of the *Seattle Manufacturing* organization.

Responsibility_ID: The ID of the responsibility that will be used to execute the API. For example, the *Receivables manager* responsibility ID.

User_ID: The ID of the user that will be used to execute the procedure, such as the ID of *sysadmin*.

Application_ID: The ID of the application module that will be used to execute the API. For example, ID of the general Ledger module.

iWay_Oraapps: Whether the procedure should be executed with oracle application logic or as a normal RDBMS procedure. Leaving this parameter blank or setting it to *Y* will execute it as an oracle application procedure and setting it to *N* will execute it as a Non-Oracle apps procedure.

iWay Oracle Limitations and Best Practices

This section lists various limitations and best practices for the iWay Application Adapter for Oracle E-Business Suite.

1. All Oracle packages are supported.
2. Oracle databases using Real Application Clusters (RAC) are supported.
3. Oracle functions are not supported.
4. XML gateway is not supported.
5. Custom data types of type Object, Table Types, and Nested Objects are supported.
6. Nested tables and tables comprised of non-scalar objects besides PL/SQL records are not supported, for example, tables comprised of Oracle VARRAYs.
7. The schema generated by the adapter has all Oracle API parameters as optional. There is no way to determine which parameters are required and which are optional from the data dictionary.
8. A package cannot consist of two or more stored procedures with the same name.
9. Stored procedures not within a package are supported. The adapter passes the same string and API_NAME parameter, and just excludes the package name while still passing two periods.

   ```
   API_NAME = 'schema_name..procedure_name'
   ```

10. It is a best practice to use the ojdbc14.jar thin driver. Other Oracle drivers provided by various third-party application vendors may cause unexpected behavior.
11. On some Oracle application installations, the sequencing in the data dictionary can be in reverse order. In these situations, the Oracle API must be recompiled.
12. The adapter supports custom APIs, but they should conform to all standard Oracle API development methodology.

13. It is a best practice to delete and archive the iway_wrapper_log.txt file periodically, as this file can grow fairly large in size.

Integration Using OAGIS XML and EDI Documents

Integrating with Open Applications Group Integration Specification (OAGIS) XML and EDI documents can be achieved with iWay touchpoint solutions and other adapters. Please contact your iWay Software representative for more information.

Deployment Information for Your iWay Adapter

Your iWay adapter works in conjunction with one of the following components:

- iWay Service Manager
- iWay Business Services Provider (iBSP)

When hosted in an iWay environment, the adapter is configured through iWay Service Manager. The iWay Transformer enables you to customize transformations to and from HL7 message format and XML. For more information on using the iWay Transformer, see *iWay Transformer User's Guide*.

iWay Explorer is used to configure system connections, create services, and configure event capabilities. When the adapter is hosted in a third-party application server environment, you can configure iWay Explorer to work in a web services environment.

iWay Service Manager

iWay Service Manager is the heart of the Universal Adapter Framework and is an open transport service bus. Service Manager uses graphical tools to create sophisticated integration services without writing custom integration code by:

- Creating metadata from target applications.
- Transforming and mapping interfaces.
- Managing stateless processes.

Its capability to manage complex adapter interactions makes it ideally suited to be the foundation of a service-oriented architecture.

iWay Transformer

iWay Transformer is a rules-based, data transformation tool that converts an input file of one data format to an output file of another data format. iWay Transformer's easy to use graphical user interface enables you to design transformation projects specific to your needs.

iWay Explorer

iWay Explorer uses a tree metaphor to introspect a system for metadata. The explorer enables you to create XML schemas and web services for the associated object. In addition, you can create ports and channels to listen for events in a system. External applications that access a system through the adapter use either XML schemas or web services to pass data between the external application and the adapter.

iWay Business Services Provider (iBSP)

The iWay Business Services Provider (iBSP) exposes, as web services, enterprise assets that are accessible from adapters regardless of the programming language or the particular operating system.

iBSP simplifies the creation and execution of web services when running:

- Custom and legacy applications.
- Database queries and stored procedures.
- Packaged applications.
- Terminal emulation and screen-based systems.
- Transactional systems.

Coupled with a platform and language independent messaging protocol called SOAP (Simple Object Access Protocol), XML enables application development and integration by assembling previously built components from multiple web services.

Application Adapter for Oracle E-Business Suite Information Roadmap

The following table lists the location of deployment and user information for components of the iWay Application Adapter for Oracle E-Business Suite.

Deployed Component	For more information, see
iWay Service Manager	Chapter 6 of this guide *iWay Service Manager User's Guide*
iWay Explorer	Chapters 3 and 4 of this guide *iWay Installation and Configuration*

Deployed Component	For more information, see
iWay Business Services Provider (iBSP)	*iWay Installation and Configuration*

Chapter 2

Application Adapter for Oracle E-Business Suite Supported Platforms Matrix

iWay Software is committed to support the diverse environments and varied systems of our users through support for leading enterprise applications, platforms, and databases.

This section specifies version, platform, and database support information for iWay Application Adapter for Oracle E-Business Suite. It is designed to provide a consolidated view of Oracle E-Business Suite releases and the various operating systems and databases, on which they are supported.

In this chapter:

Application Adapter for Oracle E-Business Suite Supported Platforms Overview

iWay Application Adapter for Oracle E-Business Suite supports Oracle E-Business Suite v11i and R12.2.2. The adapter communicates to Oracle E-Business Suite with PL/SQL API, Views, Business Events, Open Interface Tables, and Concurrent Programs using the iWay wrapper.

Supported Application Adapter for Oracle E-Business Suite Versions

iWay Application Adapter for Oracle E-Business Suite supports Oracle E-Business Suite as shown in the following list:

- Oracle E-Business Suite v11i
- Oracle E-Business Suite R12.2.2

Note: The ojdbc14.jar file is required for integration purposes.

Application Adapter for Oracle E-Business Suite Operating Systems

iWay Application Adapter for Oracle E-Business Suite supports all of the operating systems that are listed in the *iWay Installation and Configuration Guide* under *Operating System Requirements*.

Databases

iWay Application Adapter for Oracle E-Business Suite works directly with 11g R2, 12C Release1 databases, and published APIs/Packages using the iWay wrapper.

Java Development Kit (JDK)

iWay Application Adapter for Oracle E-Business Suite supports the Java Development Kit (JDK) versions that are listed in the *iWay Installation and Configuration Guide* under *Java Requirements*.

Application Adapter for Oracle E-Business Suite Communication Modes

iWay Application Adapter for Oracle E-Business Suite supports the following communication modes:

- **Services (Outbound).** iWay Application Adapter for Oracle E-Business Suite can send messages to Oracle E-Business Suite.

- **Events (Inbound).** iWay Application Adapter for Oracle E-Business Suite can receive messages from Oracle E-Business Suite.

Application Adapter for Oracle E-Business Suite Object Types and Interfaces

iWay Application Adapter for Oracle E-Business Suite supports the following Oracle E-Business Suite Types and Interfaces:

- **PL/SQL API (Packages).** Packages are used for outbound communication from Oracle E-Business Suite adapter to Oracle E-Business Suite.
- **Views.** Views are used for inbound communication from Oracle E-Business Suite to Oracle E-Business Suite adapter.
- **Business Events.** Business Events are used for inbound communication from Oracle E-Business Suite to Oracle E-Business Suite adapter.
- **Open Interface Table.** Open Interface Table are used for outbound communication from Oracle E-Business Suite adapter to Oracle E-Business Suite.
- **Concurrent Programs.** Concurrent Programs are used for outbound communication from Oracle E-Business Suite adapter to Oracle E-Business Suite.

Application Adapter for Oracle E-Business Suite Communication Types

iWay Application Adapter for Oracle E-Business Suite supports the following communication types:

- **PL/SQL API (Packages):** Synchronous
- **Concurrent Programs:** Asynchronous
- **Views:** Synchronous
- **Business Events:** Synchronous
- **Open Interface Tables:** Asynchronous

Application Adapter for Oracle E-Business Suite Operations

iWay Application Adapter for Oracle E-Business Suite supports the following operations:

- **PL/SQL API (Packages):** All the operations supported by Packages. Please note that operations vary based on each of the packages.
- **Views:** Query
- **Open Interface Table.** Insert and Query
- **Concurrent Programs.** Run
- **Business Events:** Receive and Query

Application Adapter for Oracle E-Business Suite Data Types

iWay Application Adapter for Oracle E-Business Suite supports the following data types:

- BINARY_DOUBLE
- BINARY_FLOAT
- BLOB
- CHAR
- CLOB and NCLOB
- DATE
- INTERVAL YEAR TO MONTH
- INTERVAL DAY TO SECOND
- LONG
- LONG RAW
- NCHAR
- NUMBER
- NVARCHAR2
- RAW

- TIMESTAMP
- TIMESTAMP WITH TIMEZONE
- TIMESTAMP WITH LOCAL TIMEZONE
- VARCHAR and VARCHAR2
- Custom data types of type OBJECT
- Nested tables and tables comprised of non-scalar objects

Other Application Adapter for Oracle E-Business Suite Functions

There is no known list related to other functions for iWay Application Adapter for Oracle E-Business Suite.

Application Adapter for Oracle E-Business Suite Known Limitations

This section lists known issues for iWay Application Adapter for Oracle E-Business Suite.

- Functions are not supported.
- XML gateway is not supported.
- Only custom data types of type OBJECT are supported. Other custom data types are not supported.
- Tables comprised of arrays are not supported.
- A package cannot consist of two or more stored procedures with the same name.
- The adapter supports custom APIs, but they should conform to all standard Oracle API development methodology.

Related Information for Application Adapter for Oracle E-Business Suite in Specific iWay Releases

For more information, see the *iWay New Features Bulletin and Release Notes* documentation for a specific release (for example, iWay Version 7.0.2).

Chapter 3

Application Adapter for Oracle E-Business Suite Quick Start Guide

This chapter provides a quick start guide for the iWay Application Adapter for Oracle E-Business Suite.

In this chapter:

Application Adapter for Oracle E-Business Suite Quick Start Overview

This quick start guide summarizes the high-level key steps that are required to install, configure, and use the iWay Application Adapter for Oracle E-Business Suite. The quick start guide does not elaborate on any of the steps in detail. Instead, cross-references are provided for the corresponding sections in the *iWay Application Adapter for Oracle E-Business Suite User's Guide*. Users of the iWay Application Adapter for Oracle E-Business Suite are encouraged to follow the sequence of steps in this guide to quickly connect to Oracle E-Business Suite systems and begin using the adapter. To gain a complete understanding about the adapter, it is recommended for users to review the entire *iWay Application Adapter for Oracle E-Business Suite User's Guide*, as the quick start guide section is not a replacement for that level of detail.

Oracle E-Business Suite Quick Start Guide

This section lists and describes the key configuration steps for configuring the iWay Application Adapter for Oracle E-Business Suite and then integrating with Oracle E-Business Suite.

1. Ensure that you are using a supported environment, as described in *Application Adapter for Oracle E-Business Suite Supported Platforms Matrix* on page 33.
2. Copy the *ojdbc14.jar* file to the \lib subdirectory where iWay Service Manager (iSM) is installed.
3. Install the iWay Oracle API Wrapper.

   For more information, see *Installing the iWay Oracle API Wrapper* on page 41.
4. Open iWay Integration Tools (iIT) and access the iWay Explorer tab to create a new configuration.

   For more information, see *Starting iWay Explorer* on page 48.

5. Add the iWay Application Adapter for Oracle E-Business Suite to iWay Explorer.

   For more information, see *Creating a New Target* on page 53.

6. Create and configure an adapter target to Oracle E-Business Suite.

   For more information, see *Creating a New Target* on page 53 and *Connecting to a Target* on page 62.

7. Examine the available metadata for your Oracle published APIs that you want to integrate and create corresponding XML request and response documents.

   For more information, see *Generating XML Schemas for Oracle APIs* on page 91.

8. Create an XML request document (payload) based on the generated XML request schema.

9. Create iWay Business Services (web services) based on the generated XML schema documents.

   For more information, see *Creating iWay Business Services for Oracle APIs* on page 96.

10. Test the iWay Business Service (web service) that has been created.

    For more information, see *How to Create and Test a Web Service for a Stored Procedure* on page 97.

11. Verify the request using the Oracle E-Business Suite client.

12. Create a process flow that consists of an Adapter object based on your Oracle E-Business Suite adapter target configured using iWay Explorer.

    For more information, see *Configuring the Application Adapter for Oracle E-Business Suite in iWay Integration Tools Designer* on page 197.

Chapter 4

Installing the iWay Oracle API Wrapper

If you want to use the iWay Application Adapter for Oracle E-Business Suite to interact with Oracle APIs (Stored Procedures and Packages), you must install the iWay Oracle API wrapper. The following topics provide the instructions that are needed to perform a new iWay Oracle API wrapper installation or upgrade an existing version. Detailed information about the iWay Oracle API wrapper architecture is also provided as a reference.

In this chapter:

iWay Oracle API Wrapper Prerequisites

The following prerequisites must be available before proceeding with the iWay Oracle API wrapper installation:

- All of the iWay Oracle API wrapper installation files:
  - **create_iway_tables.sql** - Drops and creates the necessary tables in the Oracle database.
  - **iway_ora_api_decl.pks** - Dynamic declaration script for Oracle API declarations.
  - **iway_wrapper_decl.pks** - iWay wrapper declaration script for get_api_definition and run_api.
  - **iway_ora_api_wrapper.pks** - The package header script for the wrapper.
  - **iway_ora_api_wrapper.pkb** - The package body script for the wrapper.
- A custom database user account with the name *iWay* must be available for the E-Business Suite database server. This account must have a DBA role and privileges to create, alter, delete, drop, grant, update tables, synonyms, and procedures.

  If the account is not available, please create one.

Granting Privileges for the APPS and IWAY Database Users

This section describes how to grant the required privileges for the APPS and IWAY database users. You must grant these privileges before you install the iWay Oracle API wrapper.

Note: These privileges must be granted only if Oracle Applications is not installed.

For the APPS database user:

```
GRANT ALTER ANY PROCEDURE TO "APPS";
GRANT ALTER ANY TABLE TO "APPS";
GRANT ALTER ANY TRIGGER TO "APPS";
GRANT ALTER DATABASE TO "APPS";
GRANT ALTER SYSTEM TO "APPS";
GRANT ALTER USER TO "APPS";
GRANT CREATE ANY PROCEDURE TO "APPS";
GRANT CREATE ANY SYNONYM TO "APPS";
GRANT CREATE ANY TABLE TO "APPS";
GRANT CREATE ANY TRIGGER TO "APPS";
GRANT CREATE PUBLIC SYNONYM TO "APPS";
GRANT DELETE ANY TABLE TO "APPS";
GRANT DROP ANY PROCEDURE TO "APPS";
GRANT DROP ANY SYNONYM TO "APPS";
GRANT DROP ANY TRIGGER TO "APPS";
GRANT DROP ANY VIEW TO "APPS";
GRANT EXECUTE ANY PROCEDURE TO "APPS";
GRANT GRANT ANY OBJECT PRIVILEGE TO "APPS";
GRANT SELECT ANY DICTIONARY TO "APPS";
GRANT SELECT ANY TABLE TO "APPS";
GRANT UNLIMITED TABLESPACE TO "APPS";
GRANT UPDATE ANY TABLE TO "APPS";
GRANT "CONNECT" TO "APPS";
GRANT "RESOURCE" TO "APPS";
```

For the IWAY database user:

```
GRANT CREATE ANY PROCEDURE TO "IWAY";
GRANT CREATE SYNONYM TO "IWAY";
GRANT CREATE ANY TABLE TO "IWAY";
GRANT CREATE PROCEDURE TO "IWAY";
GRANT CREATE TABLE TO "IWAY";
GRANT CREATE TABLESPACE TO "IWAY";
GRANT CREATE TRIGGER TO "IWAY";
GRANT DROP ANY PROCEDURE TO "IWAY";
GRANT DROP ANY TABLE TO "IWAY";
GRANT DROP ANY VIEW TO "IWAY";
GRANT DROP PUBLIC SYNONYM TO "IWAY";
GRANT DROP TABLESPACE TO "IWAY";
GRANT EXECUTE ANY PROCEDURE TO "IWAY";
GRANT GRANT ANY OBJECT PRIVILEGE TO "IWAY";
GRANT GRANT ANY PRIVILEGE TO "IWAY";
GRANT INSERT ANY TABLE TO "IWAY";
GRANT SELECT ANY TABLE TO "IWAY";
GRANT UNLIMITED TABLESPACE TO "IWAY";
GRANT UPDATE ANY TABLE TO "IWAY";
GRANT "CONNECT" TO "IWAY";
GRANT "EXECUTE_CATALOG_ROLE" TO "IWAY";
```

Installing the GETCLOB Procedure

The getclob procedure must be installed under the schema the adapter uses to run Oracle APIs. To install the getclob procedure:

1. Extract the iwse.ora script from the ibspsql.zip archive, which is located in the following directory:

   ```
   <iway_home>\etc\setup
   ```

2. Run the iwse.ora script using SQL*PLUS.

Installing the iWay Oracle Wrapper

Once you have reviewed the prerequisites, you can begin the iWay Oracle API wrapper installation.

Note: It is recommended that all scripts are run using SQL*PLUS.

Procedure: How to Install the iWay Oracle Wrapper

To install the iWay Oracle wrapper:

1. Log into the Windows or UNIX environment where the Oracle E Business suite software is installed.
2. Locate the *.pks, *.pkb, and *.sql files in the following directory:

   iwaySMHome\etc\setup

where:

iWaySMHome

Is the location where iWay 7.0 SM is installed.

3. Copy the *.pks, *.pkb, and *.sql files into the $IWAY_TOP/admin/sql directory.

   If this directory is not available, you can copy the files to any other custom directory in the Oracle E-Business Suite environment.

4. Change directory to $IWAY_TOP/admin/sql (or to the directory where you copied the wrapper code files), so that it is the current working directory.

5. Log into SQL*Plus as *iWay* database user.

6. Run the following scripts to create the IWAY_API_PARAMETERS table, IWAY_DECLARATIONS table, and the IWAY_OUTPUT_STRINGS temporary table:

```
SQL> SPOOL IWAY_DECL_TABLE_INSTALL_LOG.txt
SQL> @create_iway_tables.sql
SQL> spool off
```

   Note: This script drops existing tables with the target name "iway_api_parameters", "iway_declarations", and "iway_output_strings", creates the tables, and grants access to apps. If this is a new install, the first part of the script will fail (drops the object) but the table creation and the grant should succeed. This is expected behavior.

7. **For New Installations Only.** This step is not required for upgrades because the output directory is already set. If you are performing an upgrade, continue to Step 8.

   Log into SQL*Plus as *APPS* user.

   Note: If *APPS* user does not exist in non-Oracle application environments, you must create a user called APPS with a DBA role and privileges to create, alter, delete, drop, grant, update tables, synonyms, and procedures.

   Verify the following:

```
select num, name, value
from v$parameter
where name = 'utl_file_dir';
```

   If the result of the query is not '/usr/tmp', then change the following declaration line in the file IWAY_ORA_API_WRAPPER.pks to point to a valid directory as shown in the value column from the above query result, and save the file.

   gv_log_file_dir VARCHAR2 (100) := '/usr/tmp';

   The current value '/usr/tmp' can be left unchanged if the directory path is configured in the utl_file_directory database parameter. This is the location where the debug file will be created if debugging is enabled.

If the utl_file_dir parameter is not configured on the database system, run the following command:

```
alter system set utl_file_dir=value scope=spfile;
```

where:

value

Is the full path you want to configure (for example, D:\oracle\visdb\).

This command configures the utl_file_dir parameter, which is what you need to enter in the gv_log_file_dir parameter. This is where the wrapper log will be written.

8. Log into SQL*Plus as *APPS* user and run the following scripts and commands as follows to create PL/SQL wrapper packages.
   a. SQL> SPOOL IWAY_WRAPPER_INSTALL_LOG.txt
   b. SQL> start IWAY_WRAPPER_DECL.pks
   c. **For New Installations Only.** SQL> start IWAY_ORA_API_WRAPPER.pks
   d. **For New Installations and Upgrades.** SQL> start IWAY_ORA_API_DECL.pks
   e. Run one of the following:

      If you are installing in a database where the Oracle E-Business Suite (for example, Oracle Apps) is installed, run the following command:

      ```
      SQL> start IWAY_ORA_API_WRAPPER.pkb
      ```

      If you are installing in a database where the Oracle E-Business Suite (for example, Oracle Apps) is not installed, then run the following command:

      ```
      SQL> start IWAY_ORA_API_WRAPPER_NO_APPS.pkb
      ```

      Troubleshooting Notes: All of the scripts should complete successfully. If the scripts fail, ensure that the files are copied into the correct directory and they are available in your current working directory. If there are compilation issues, contact iWay Software Customer Support Services and provide the spool file, IWAY_WRAPPER_INSTALL_LOG.txt, that is generated during the installation for troubleshooting.
   f. GRANT EXECUTE ON iway_ora_api_wrapper TO IWAY WITH GRANT OPTION;
   g. GRANT EXECUTE ON iway_wrapper_decl TO IWAY WITH GRANT OPTION;
   h. CONNECT *iway* (This changes accounts from APPS to IWAY.)
   i. When prompted, enter the password for the iWay account.

j. DROP SYNONYM iway_ora_api_wrapper;

Note: This will fail if the synonym does not exist. This is the expected behavior.

k. CREATE SYNONYM iway_ora_api_wrapper FOR APPS.iway_ora_api_wrapper;

l. SQL> SPOOL off.

9. If this is not a fresh install, log into SQL*Plus as APPS and issue the following:

```
SQL> set heading off
SQL> set echo off
SQL> set termout off
SQL> spool drop_decl.sql
SQL> SELECT 'DROP PACKAGE ' || OBJECT_NAME || ';' FROM DBA_OBJECTS
WHERE OBJECT_NAME LIKE 'IWAY2%';
SQL> spool off
SQL> @drop_decl.sql
```

10. Log off from SQL*Plus to complete the installation process.

Chapter 5

Configuring and Managing Connections to Oracle E-Business Suite

This section describes how to configure and manage connections to Oracle E-Business Suite using iWay Explorer.

In this chapter:

Oracle E-Business Suite Connections Overview

You can use iWay Explorer to:

- Establish a connection to Oracle E-Business Suite.
- View metadata that describes Oracle interface tables or custom interface tables. You can use this metadata when you create request documents and when you develop logic for processing response documents.
- Search for stored procedures under a package.
- Interact directly with Oracle base tables and views. This integration method is most appropriate when you need to query data located in Oracle base tables and views.
- Generate XML schemas that define request and response documents for Oracle interface tables or custom interface tables, stored procedures under a package, or base tables and views. You can use these schemas when you create request documents and when you develop logic for processing response documents.
- Create business services (also known as web services) for your interface tables, stored procedures under a package, or base tables and views.
- Create iterative batch processes.

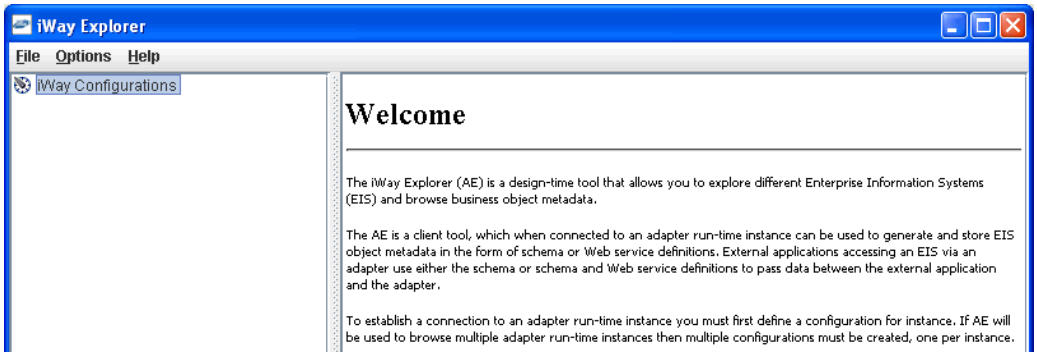
- Create parameterized SQL statements.

Starting iWay Explorer

This topic describes how to start iWay Explorer.

- If you are running iWay Explorer on Windows, see *How to Start iWay Explorer on Windows* on page 48.
- If you are running iWay Explorer on UNIX platforms with an X Server, see *How to Start iWay Explorer on UNIX Platforms With an X Server* on page 49.

The following image shows the iWay Explorer window that opens after iWay Explorer loads. A navigation pane is on the left. The navigation pane contains the root node, iWay Configurations.



From this window, you can create a new configuration for iWay Adapters and a new target for the iWay Application Adapter for Oracle E-Business Suite.

Procedure: How to Start iWay Explorer on Windows

To start iWay Explorer on Windows:

1. Ensure iWay Explorer is installed.

   For more information on installing and configuring iWay Explorer, see the *iWay Installation and Configuration* documentation.

2. From the Windows Start menu, select *Programs*, *iWay 7.0 Service Manager*, *tools*, and click *iWay Explorer - SWING*.

   A status window indicates the loading progress of iWay Explorer. The iWay Explorer window then opens.

Procedure: How to Start iWay Explorer on UNIX Platforms With an X Server

To start iWay Explorer on UNIX platforms with an X Server:

1. Ensure iWay Explorer is installed.

   For more information on installing and configuring iWay Explorer, see the *iWay Installation and Configuration* documentation.

2. Execute the following command:

   ```
   /opt/iWay7/tools/iwae/bin/iwae.sh
   ```

   A status window indicates the loading progress of iWay Explorer. The iWay Explorer window then opens.

Creating a New Configuration

Before you can use iWay Explorer with iWay adapters, you must create a repository where your XML schemas, web services, and event data are stored. Since you can deploy iWay Explorer using the iWay Business Services Provider (iBSP), each implementation requires you to configure a specific repository before you can explore Enterprise Information System (EIS) metadata.

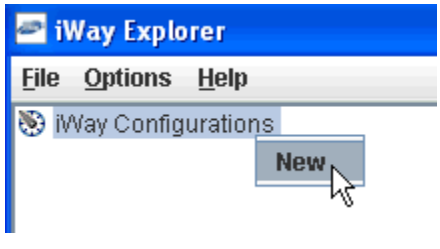
The iBSP exposes, as web services, enterprise assets that are accessible from adapters regardless of the programming language or the particular operating system. In addition, you can use iBSP as a stand-alone Java application running in iWay Service Manager. If you deployed iWay Explorer using iBSP, follow the steps in *How to Create a Repository for iBSP* on page 49.

Procedure: How to Create a Repository for iBSP

To create a repository for iBSP using iWay Explorer:

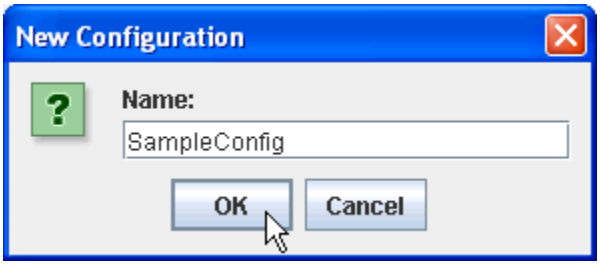
1. Start iWay Explorer as described in *Starting iWay Explorer* on page 48.

   The following image shows the New option that is available in the left pane of the iWay Explorer window when you right-click the iWay Configurations node.



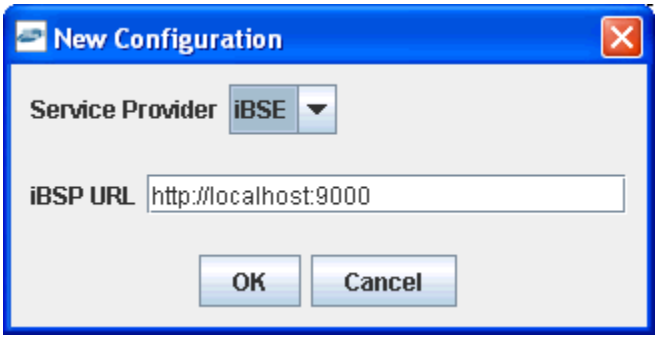
2. Right-click *iWay Configurations* and select *New*.

   The following image shows the New Configuration dialog box that opens, where you supply a name for the configuration.



3. Type a name for the configuration, for example, SampleConfig, and click *OK*.

   The following image shows additional fields on the New Configuration dialog box, where you select the service provider and supply the iBSP URL.



4. From the Service Provider drop-down list, select *iBSE*.

   You are prompted for the iBSP URL. Use one of the following two options:

   To access the stand-alone iBSP that is installed as part of iWay Service Manager, provide the following URL

   `http://hostname:9000`

   where:

   hostname

   Is the host name where iWay is installed.

   If you changed the default SOAP port, substitute accordingly.

   To access Servlet iBSP, provide the following URL

```
http://hostname:port/ibse/IBSEServlet
```

where:

hostname

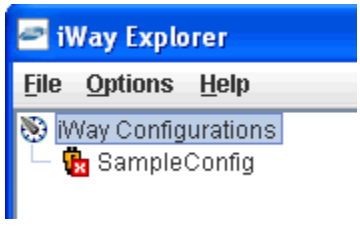
Is the host name of your application server.

port

Is the port number used by your application server.

5. Click *OK*.

The following image shows the node representing the new configuration that appears beneath the root iWay Configurations node. In this image, the node is named SampleConfig.



The adapters are installed with the iWay installation. However, you must provide the appropriate third-party .JAR files to display each adapter in the list of available adapters in the left pane. For more information, see the *iWay Installation and Configuration Guide*.

The iWay Adapters enable you to view EIS metadata and create XML request and response schemas that can be used to listen for events or create web services. For more information on viewing EIS metadata and creating schemas, see *Integrating With Oracle Interface Tables and Concurrent Programs* on page 67. For more information on creating web services, see *Creating iWay Business Services for Interface Tables* on page 74.

The iWay Events enable you to create ports and channels for event handling. For more information, see *Listening for Oracle E-Business Suite Events* on page 153.

Connecting to a New Configuration

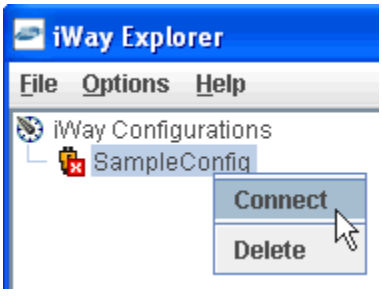
Once you have created a new configuration using iWay Explorer, you must connect to it before you can use your iWay Application Adapter for Oracle E-Business Suite.

Procedure: How to Connect to a New Configuration

To connect to a new configuration:

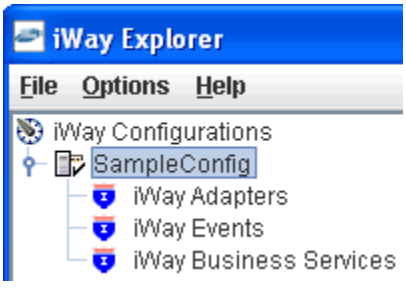
1. Right-click the configuration to which you want to connect, for example, SampleConfig.

   The following image shows the options (Connect and Delete) that are available in the left pane of the iWay Explorer window when you right-click the SampleConfig node.



2. Select *Connect*.

   The following image shows the left pane, where nodes appear for iWay Adapters, iWay Events, and iWay Business Services (also known as web services).



   Use the iWay Adapters node to create inbound interaction with an EIS. For example, you use the Oracle node in the iWay Adapters node to configure a service that updates Oracle E-Business Suite.

   Use the iWay Events node to configure listeners that listen for events in Oracle E-Business Suite.

   Use the iWay Business Services node to test business services created in the iWay Adapters node. You can also control security settings for the business services that are available.

Connecting to Oracle E-Business Suite

To browse and work with Oracle E-Business Suite metadata, you must create a target for Oracle E-Business Suite. This target serves as your connection point. You must establish a connection to Oracle every time you start iWay Explorer or after you disconnect from the system.

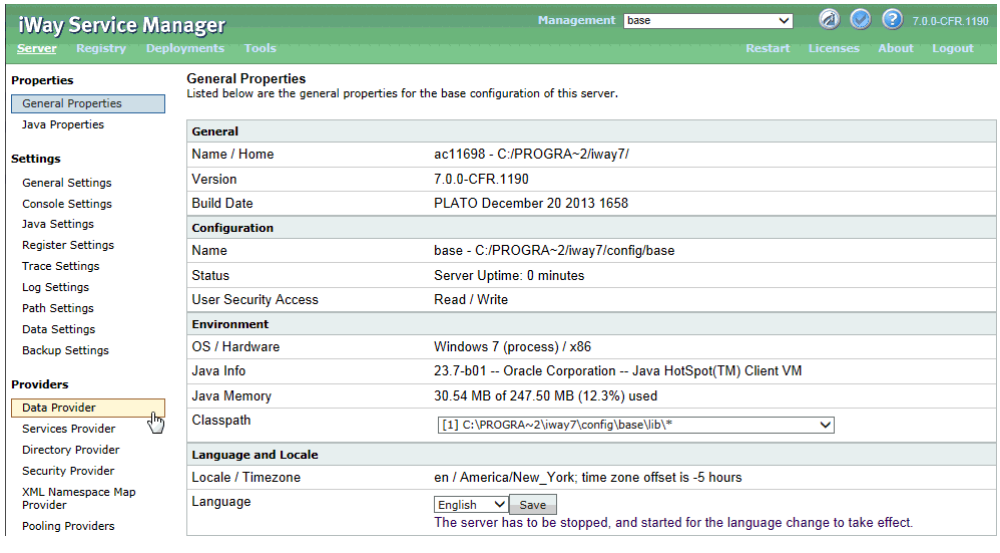
Creating a New Target

A target serves as the connection point to your Enterprise Information System (EIS) and is automatically saved after you create it. To connect to Oracle E-Business Suite for the first time, you must create a new target. Before a target is created in iWay Explorer, you must define a Data Provider using the iWay Service Manager Administration Console.

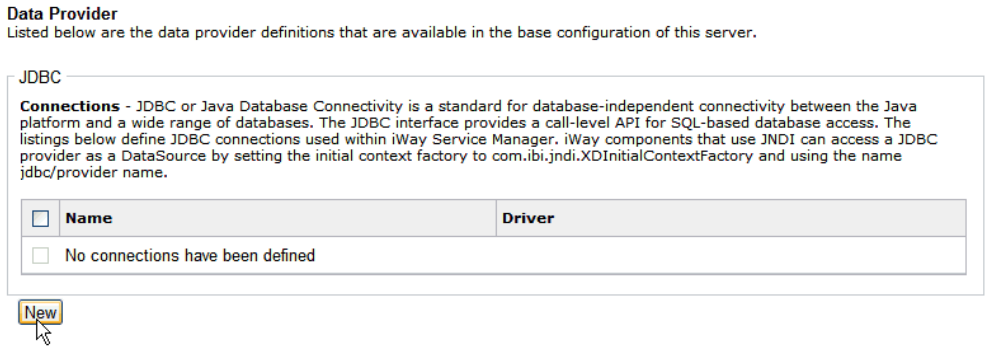
Procedure: How to Define a Data Provider Using the iWay Service Manager Administration Console

To define a Data Provider using the iWay Service Manager Administration Console:

1. Ensure iWay Service Manager (iSM) is started.
2. Log on to the iWay Service Manager Administration Console.
3. Click *Data Provider* under the Providers section in the left pane, as shown in the following image.

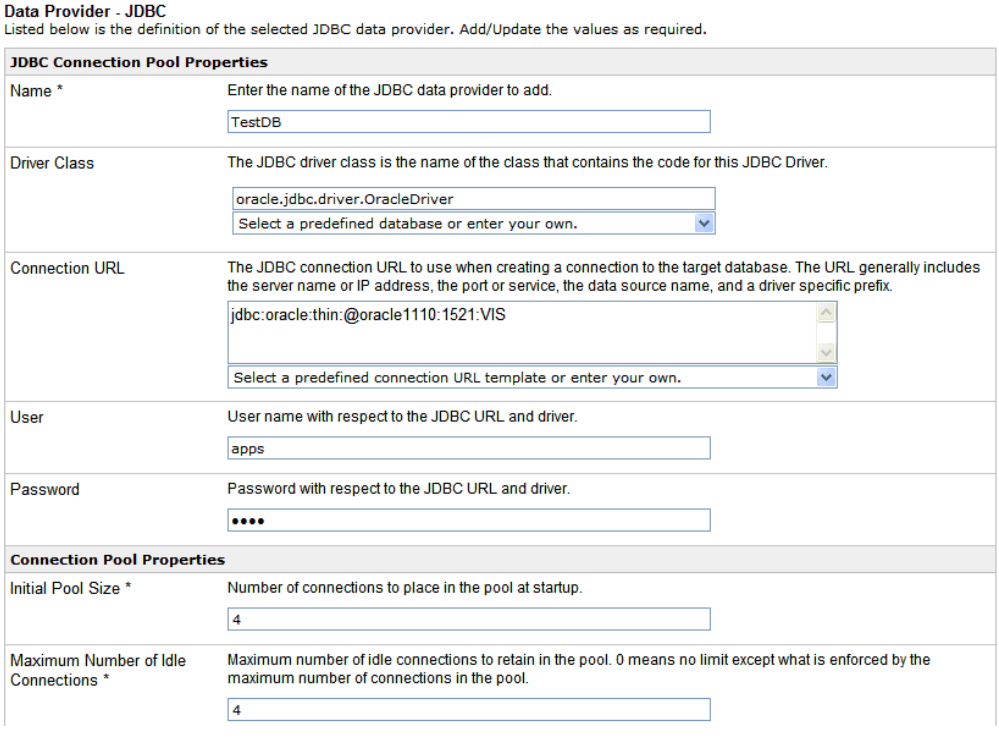


The Data Provider pane opens, as shown in the following image.



4. Click *New* In the JDBC section.

The Data Provider - JDBC pane opens, as shown in the following image.



The following table lists and describes the parameters in the Data Provider - JDBC pane that must be configured for the data provider that you are defining.

Parameter	Description
JDBC Connection Pool Properties	
Name *	Enter the name of the JDBC data provider to add (for example, TestDB).
Driver Class	The JDBC driver class is the name of the class that contains the code for this JDBC Driver. Select the following predefined driver class from the drop-down list: `Oracle Thin - {oracle.jdbc.driver.OracleDriver}` The required .jar files for the JDBC driver must be installed and registered in the iSM classpath. You can use the Path Settings option on the console to add Java classes and libraries.

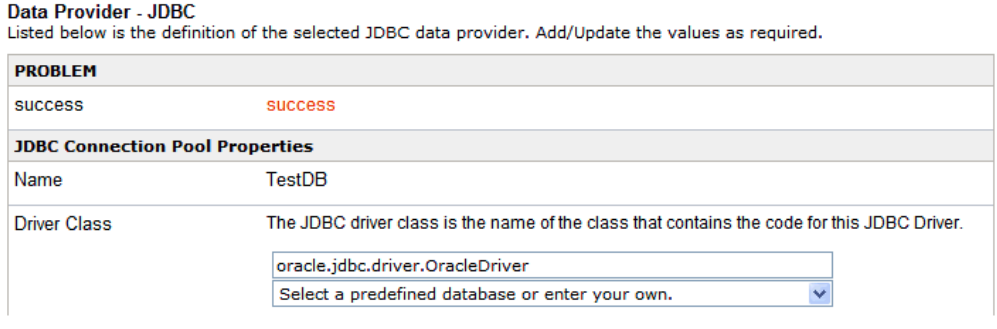
Connection URL	The JDBC connection URL to use when creating a connection to the target database. The URL generally includes the server name or IP address, the port or service, the data source name, and a driver specific prefix. Select the following predefined driver class from the drop-down list:
	`Oracle Thin - jdbc:oracle:thin:@[HOST][:PORT]:[SID]`
	The following is a sample value for the URL:
	`jdbc:oracle:thin:@oracle1110:1521:VIS`
	For more information, see the JDBC documentation for the specific data source.

User	User name with respect to the JDBC URL and driver. SREG names can also be used.
Password	Password with respect to the JDBC URL and driver. SREG names can also be used.
Connection Pool Properties	
Initial Pool Size *	Number of connections to place in the pool at startup.

Parameter	Description
Maximum Number of Idle Connections *	Maximum number of idle connections to retain in the pool. A value of 0 means no limit except what is enforced by the maximum number of connections in the pool.
Maximum Number of Connections *	Maximum number of connections in the pool. A value of 0 means no limit.
Login Timeout	Time in seconds to wait for a pooled connection before throwing an exception. A value of 0 means no limit.
Behavior When Exhausted	What to do when the pool reaches the maximum number of connections. Block means wait for a connection to become available for the period defined by the login timeout parameter. Fail means throw an exception immediately.
Validation SQL	SQL statement that can be executed to validate the health of a pooled connection. The statement should return a result set of at least one row.
Validate on Borrow	If set to *true*, the validation SQL statement will be executed on a pooled connection before returning the connection to the caller.
Validate on Return	If set to *true*, the validation SQL statement will be executed on a pooled connection before replacing the connection in the pool.

5. Click *Test* when you have entered all of the required parameters for your data provider.

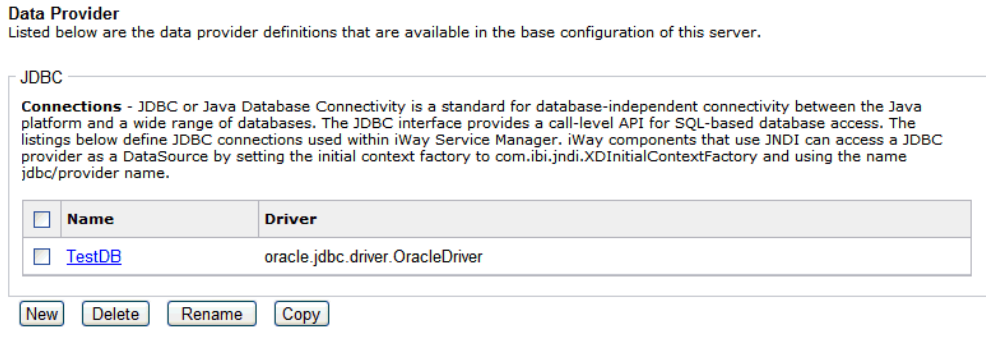
   A success message is displayed at the top of the pane if the data provider has been configured correctly, as shown in the following image.



If a connection cannot be made, an error message displays describing the problem. Typically, the driver has not been installed or the classpath has not been set.

6. Click *Add*.

   You are returned to the Data Provider pane, where the new JDBC data provider you defined (for example, TestDB) is added to the list of available providers, as shown in the following image.



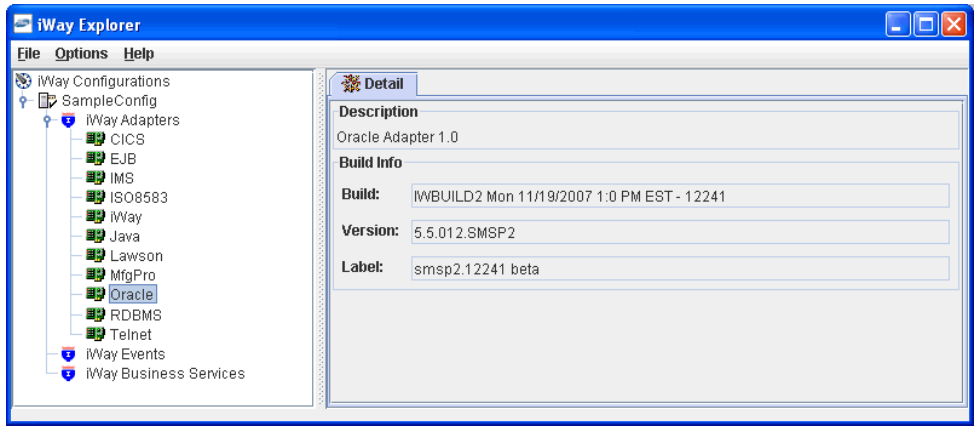
   You can now define a data source target for the adapter using iWay Explorer.

Procedure: How to Create a New Target Using iWay Explorer

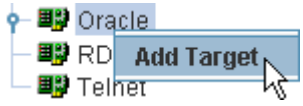
To create a new target using iWay Explorer:

1. In the left pane, expand the *iWay Adapters* node and select the *Oracle* node.

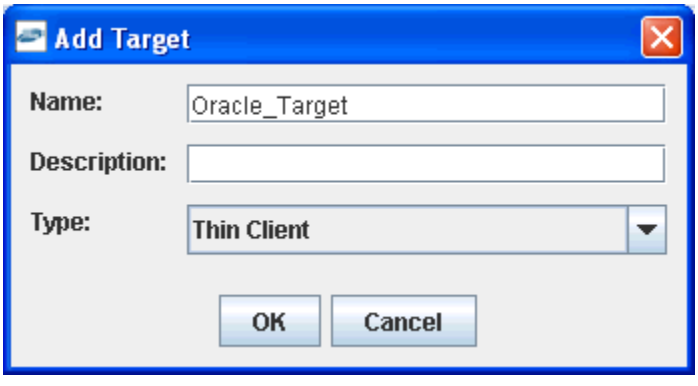
   The following image shows the list of supported adapters in the left pane and information about the selected adapter on the right.



2. Right-click the Oracle adapter node and select Add Target, as shown in the following image.



The Add Target dialog box opens, as shown in the following image. This dialog box provides fields to define the new target.



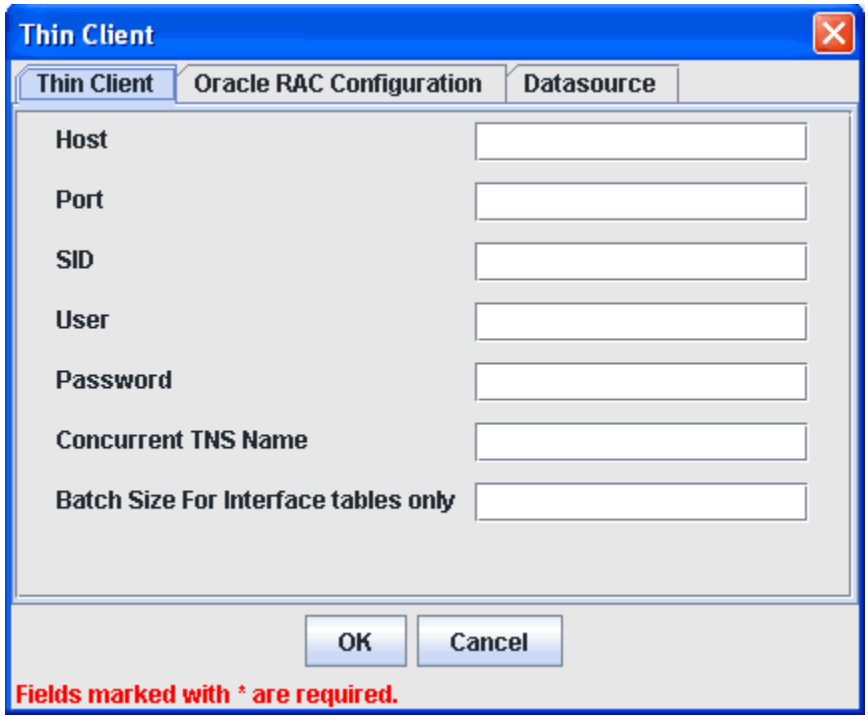
a. In the Name field, type a descriptive name for the target, for example, Oracle_Target.

b. In the Description field, type a brief description of the connection (optional).

c. From the Type drop-down list, select *Thin Client* (default).

3. Click *OK*.

The Thin Client dialog box opens.



Note: The connection parameters are consistent with those found in your Oracle E-Business Suite system. For more information on parameter values that are specific to your Oracle E-Business Suite configuration, consult your Oracle E-Business Suite system administrator.

The following table lists and describes the connection parameters for a Thin Client target.

Parameter	Description
Thin Client Tab	
Host	Name of the server on which the Oracle E-Business Suite database instance resides.
Port	Port number on which the database is listening.
SID	Unique name of the database service, chosen by the database administrator or the person who installed Oracle E-Business Suite.

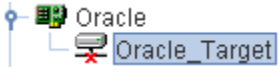
Parameter	Description
User	Oracle database user ID to access the Oracle database underlying the Oracle E-Business Suite system. The user ID must have database access to the interface tables being accessed.
Password	Password associated with the specified user ID.
Concurrent TNS Name	Logical name of the Oracle E-Business Suite database instance. (Can be obtained from the database administrator.) The TNS name that is found in the tnsnames.ora file that contains an entry for that database and that resides on the Oracle E-Business Suite database server. This is *not* the TNS name that is registered on the server on which the iWay Application Adapter for Oracle E-Business Suite is running (unless the adapter and the Oracle database reside on the same server). If a TNS name does not exist, you must create it. For more information about creating a TNS name, see your Oracle administrator.
Batch Size For Interface tables Only	The number of request document records that the adapter buffers before inserting them into an Oracle E-Business Suite interface table. When the adapter reaches the end of the request document, it inserts any remaining buffered records. The adapter tracks this batch limit separately for each interface table, not aggregately for all interface tables. For example, if you set the batch size property to 25, and the adapter accumulates 18 input records for the GL_INTERFACE table and 25 for the BUDGET_INTERFACE table, the adapter continues to hold records for GL_INTERFACE but inserts the records for BUDGET_INTERFACE. The batch size property enables you to optimize I/O: inserting too few records at a time increases I/O overhead, and inserting too many at a time ties up database resources. **Note:** The Batch size parameter is only supported for integration with Oracle E-Business Suite interface tables.

Parameter	Description
Oracle RAC Configuration Tab Enables you to configure Oracle Real Application Clusters (RAC), which provides clustering and higher availability for an Oracle RDBMS.	
Connection Cache Name	The connection name you want to use for the Oracle RAC configuration. You can use any name.
TCP Timeout	The number of milliseconds before the connection times out.
Connection Descriptor of LDAP URL	The connection description URL, which is found in the tnsnames.ora file of the Oracle RAC configuration.
Oracle Name Server (ONS)	The node setting for each Oracle system in the cluster, which includes host and port number. For example: `nodes=orac1:6200,orac2:6200` where: *orac1* and *orac2* Are the host names.
Datasource Tab	
Initial Context	Specify the following JNDI context that is provided by the JNDI service provider. `com.ibi.jndi.XDInitialContextFactory`

Parameter	Description
URL	For a JDBC connection, the JDBC driver-specific URL used to connect to the Oracle database. Specify this value using the following format `jdbc/`*`DataProviderName`* where: *`DataProviderName`* Is the JDBC data provider you defined using the iWay Service Manager Administration Console. For example: `jdbc/TestDB`
JNDI Name	JNDI name of a queue to which events are emitted. Specify this value using the following format `jdbc/`*`DataProviderName`* where: *`DataProviderName`* Is the JDBC data provider you defined using the iWay Service Manager Administration Console. For example: `jdbc/TestDB`

4. Enter values for the connection parameters according to your target type.
5. Click *OK*.

   The new target, for example, Oracle_Target, appears in the left pane beneath the Oracle node, as shown in the following image.



You can now connect to the Oracle application target you defined.

Connecting to a Target

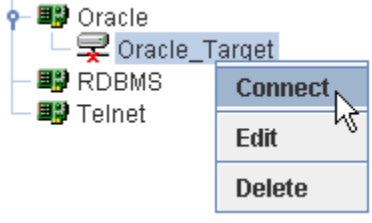
You must use a defined target to connect to an instance of Oracle E-Business Suite.

Procedure: How to Connect to a Target

To connect to an existing target:

1. In the left pane, expand the *Oracle* node and select the target you defined, for example, Oracle_Target.
2. In the right pane, enter the password for Oracle E-Business Suite.
3. Right-click the target.

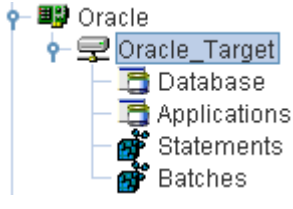
   The following image shows the options (Connect, Edit, and Delete) that are available in the left pane when you right-click the *Oracle_Target* node.



4. Select *Connect*.

   In the left pane, the Oracle_Target node changes to reflect that a connection was made.
5. Expand the target node to reveal the business objects for the Oracle E-Business Suite.

   The following image shows the Oracle_Target node expanded to reveal the Database, Applications, Statements, and Batches nodes.



Modifying, Closing, or Removing a Target

After you create a target for an EIS using iWay Explorer, you can edit the information that you provided when you created the target. For instructions on editing a target, see *How to Edit a Target* on page 64.

Although you can maintain multiple open connections to different application systems, it is recommended that you close connections when they are not in use. For instructions on closing a connection, see *How to Disconnect From a Target* on page 65.

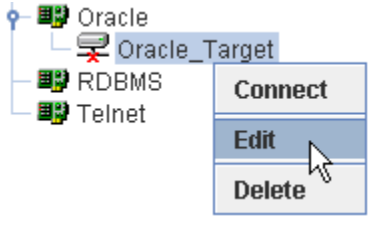
Disconnecting from a target closes the connection, but the target node remains. For instructions on deleting a target, see *How to Delete a Target* on page 65.

Procedure: How to Edit a Target

To edit a target:

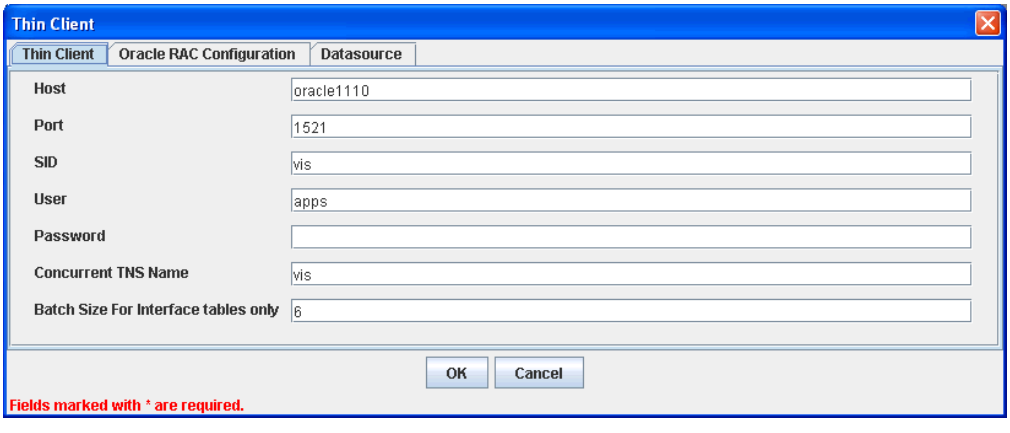
1. In the left pane, right-click the target, for example, Oracle_Target.

   The following image shows the options (Connect, Edit, and Delete) that are available in the left pane when you right-click the Oracle_Target node.



2. Select *Edit*.

   The Thin Client dialog box opens, as shown in the following image. There are three tabs: Thin Client, Oracle RAC Configuration, and Datasource. In this example, the Thin Client tab is selected and its associated fields are displayed. Use this dialog box to update the connection information to Oracle and the application server you are using.



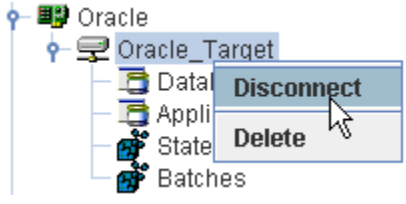
3. Modify the connection information and click *OK*.

Procedure: How to Disconnect From a Target

To disconnect from a target:

1. In the left pane, right-click the target to which you are connected, for example, Oracle_Target.

   The following image shows the Disconnect option that is available in the left pane when you right-click the *Oracle_Target* node.



2. Select *Disconnect*.

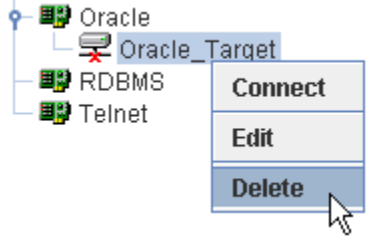
   The Oracle_Target node changes to reflect that the connection was terminated.

Procedure: How to Delete a Target

To delete a target:

1. In the left pane, right-click the target, for example, *Oracle_Target*.

   The following image shows the options (Connect, Edit, and Delete) that are available in the left pane when you right-click the *Oracle_Target* node.



2. Select *Delete*.

   The Oracle_Target node is removed from the left pane.

Chapter 6

Integrating With Oracle Interface Tables and Concurrent Programs

This section describes how to browse metadata, generate XML schemas, and create iWay Business Services for Oracle interface tables and concurrent programs.

Although this section shows the Java Swing implementation of iWay Explorer, other implementations provide the same functionality by means of similar graphical user interfaces.

In this chapter:

Browsing Interface Table Metadata

Browsing interface table metadata in iWay Explorer can be useful when creating request documents.

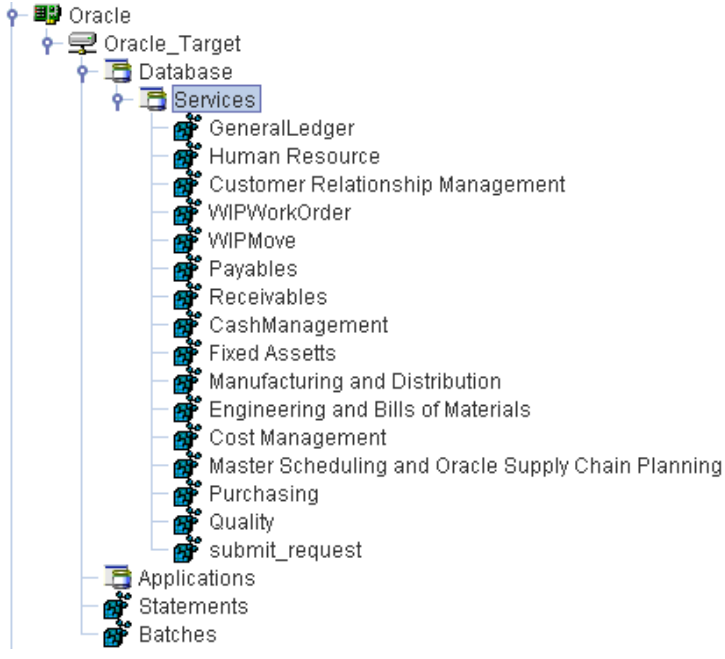
In this release, iWay Application Adapter for Oracle E-Business Suite supports two new categories of interface tables: *Human Resources (HR)* and *Customer Relationship Management (CRM)*. CRM is a new feature in Oracle Applications Release 11.5.10.

Procedure: How to Browse Interface Table Metadata

To browse Oracle E-Business Suite interface table metadata:

1. Connect to an Oracle E-Business Suite target, as described in *Connecting to Oracle E-Business Suite* on page 53.
2. In the left pane, expand the target node.

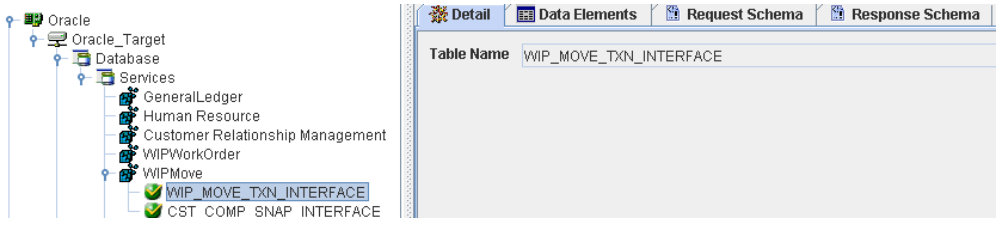
Interface table metadata is located under the Database node.



3. Expand *Database*, then the *Services* node, and then select a table that contains the metadata you want to view, for example, WIPMove.

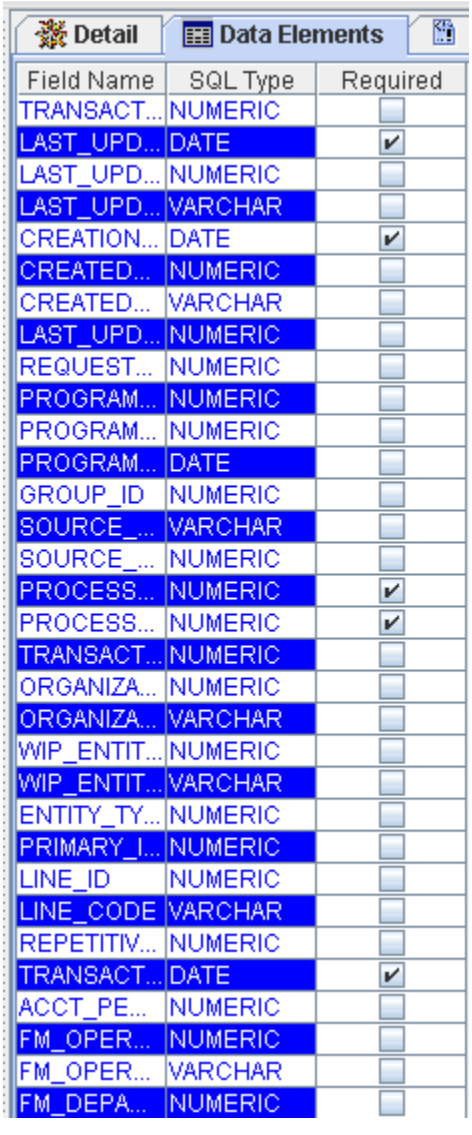
   Note: The user ID for this target connection must have read access to the selected table; otherwise you cannot view its metadata.

   When you select a specific table on the left, for example, WIP_MOVE_TXN_INTERFACE, the Detail, Data Elements, Request Schema, and Response Schema tabs appear on the right.



4. Click the *Data Elements* tab.

A detailed table appears on the right. The following image shows an example of an interface table that includes columns for the field name, the SQL data type, and for indicating whether the parameter is required.



You can use this information to determine which tables and fields you want to use when creating an XML request document or iWay Business Service.

For more information on supported interface tables, see *Supported Interface Tables for Oracle Release Apps 11i* on page 209.

Generating XML Schemas for Interface Tables

When you deploy the iWay Application Adapter for Oracle E-Business Suite in the iWay Business Services Provider (iBSP) environment, you can generate schemas that define a service request document and the corresponding response document.

If you plan to deploy the iWay Application Adapter for Oracle E-Business Suite in a business services environment, you are not required to generate a schema.

Schema Location

By default, iWay Explorer stores the schemas it creates in subdirectories under the iWay home directory of the machine where it is installed. However, you can also export these schemas using iWay Explorer to any location you specify on your file system.

The exact location of the schemas differs depending on whether you deploy iWay Explorer with an iBSP configuration.

When the adapter is used with an iBSP configuration, iWay Explorer stores the schemas in a subdirectory of the iWay installation directory, for example,

iWayHome\config\base\wsdl\schemas\service\Oracle*Oracle_Target*

where:

Oracle_Target

Is the name of the connection (target) to the Oracle Applications system that you defined using iWay Explorer. Under this directory, iWay Explorer creates subdirectories containing schemas.

You can generate schemas for interface tables, stored procedures under a package, and base tables and views. The procedure for generating a schema is identical for all three integration methods. The following procedures show examples of actual schemas generated for each integration method.

Procedure: How to Generate a Schema for an Interface Table

To generate a schema for an Oracle E-Business Suite interface table using iWay Explorer:

1. Connect to an Oracle E-Business Suite target, as described in *Connecting to Oracle E-Business Suite* on page 53.

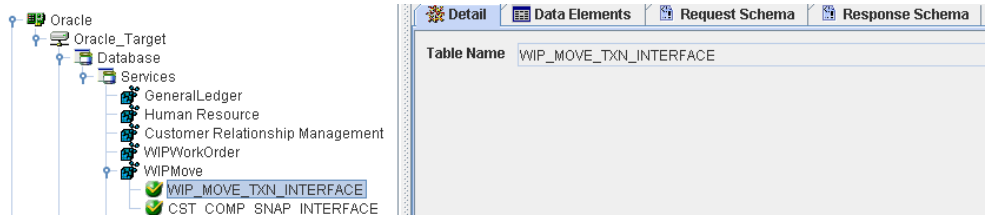
2. Locate an interface table, as described in *Browsing Interface Table Metadata* on page 67.

   Note: The user ID for this target connection must have read access to the selected table; otherwise you cannot create schemas.

3. Select a specific table on the left, for example, WIP_MOVE_TXN_INTERFACE.

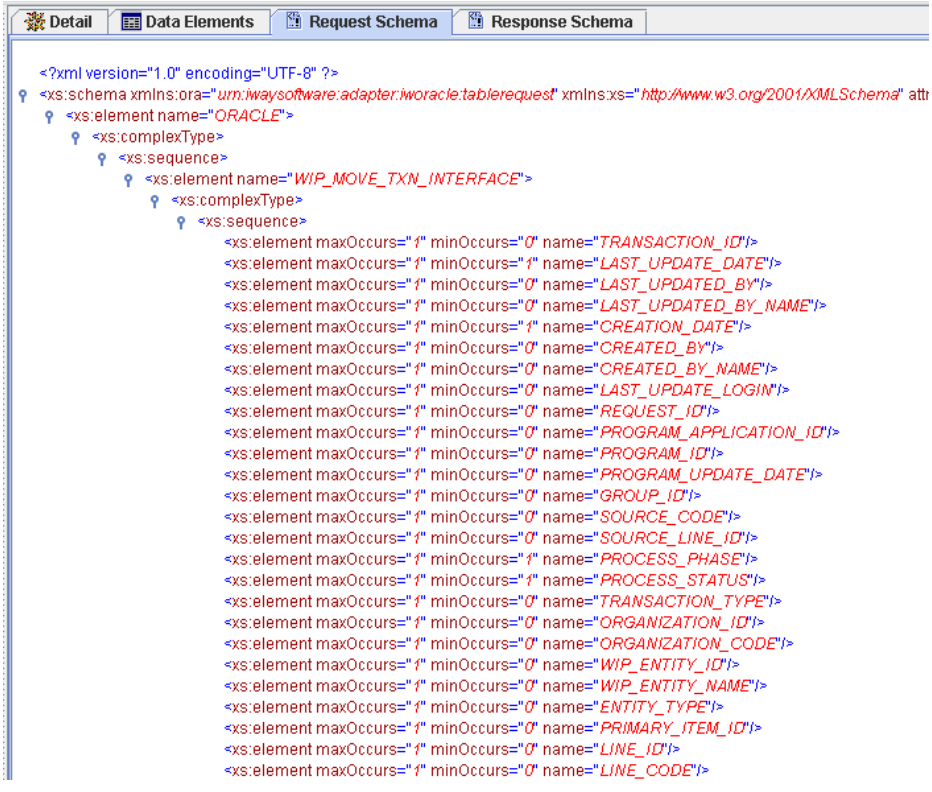
   XML request and response schemas for the table are automatically created and displayed.

   The Detail, Data Elements, Request Schema, and Response Schema tabs appear on the right.



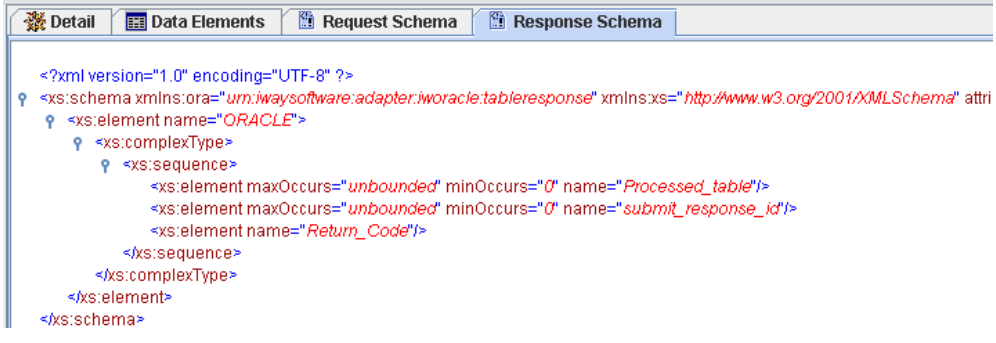
4. Click the *Request Schema* tab in the right pane.

The XML request schema appears in the right pane, as shown in the following image:

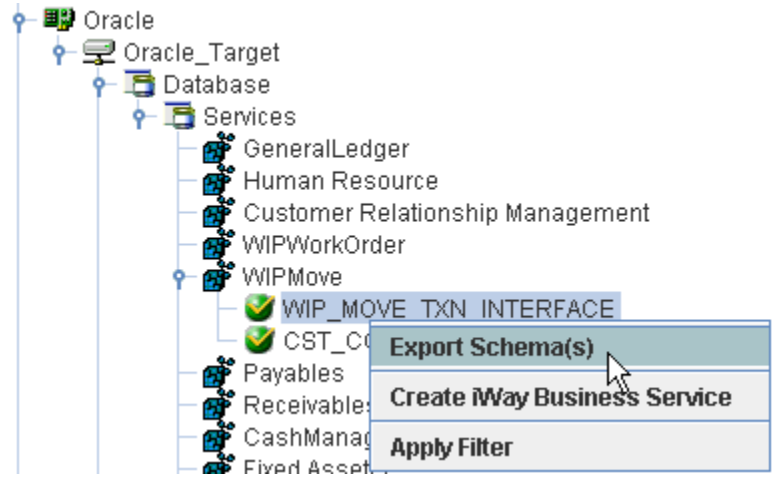


5. Click the *Response Schema* tab in the right pane.

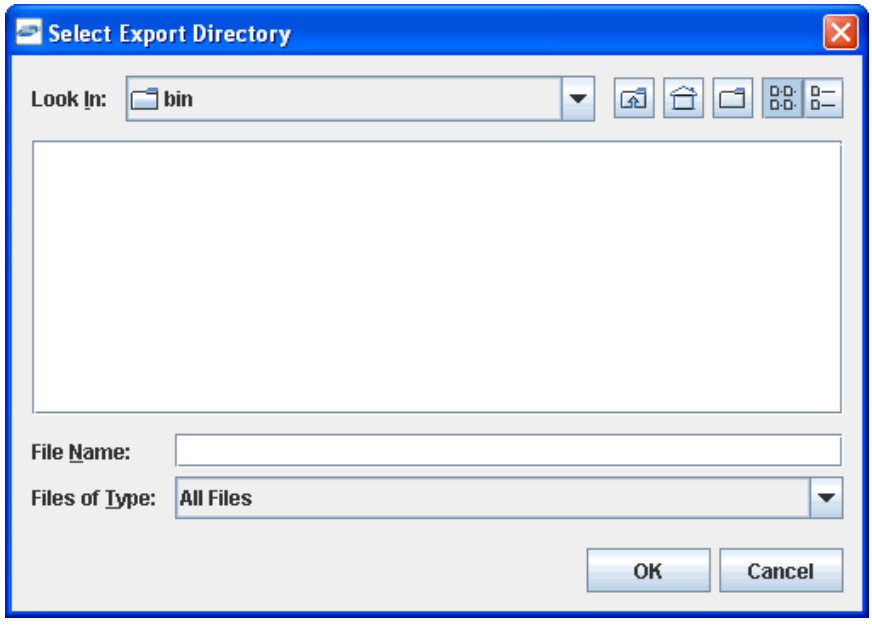
   The XML response schema appears in the right pane, as shown in the following image:



6. To export XML schemas, right-click the table in the left pane, and select *Export Schemas*, as shown in the following image:



The Select Export Directory dialog box opens, as shown in the following image:

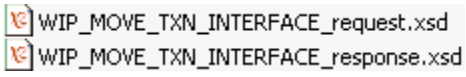


7. Navigate to a directory on your file system where you want to export the XML schemas.

   The file path displays in the File Name field.

8. Click *OK*.

The XML request and response schemas are now exported to your local file system.



Creating iWay Business Services for Interface Tables

You can generate an iWay Business Service (also known as a web service) for an Oracle E-Business Suite interface table. To generate an iWay Business Service, you must deploy the iWay Application Adapter for Oracle E-Business Suite in a business services environment using the iWay Business Services Provider (iBSP). iBSP exposes functionality as web services and serves as a gateway to heterogeneous back-end applications and databases.

A web service is a self-contained, modularized function that can be published and accessed across a network using open standards. It is the implementation of an interface by a component and is an executable entity. For the caller or sender, a web service can be considered a "black box" that may require input and delivers a result. Web services can be integrated within an enterprise as well as across enterprises on any communication technology stack, whether asynchronous or synchronous, in any format.

You can make a web service available to other services within a host server by generating WSDL (Web Services Description Language) from the web service.

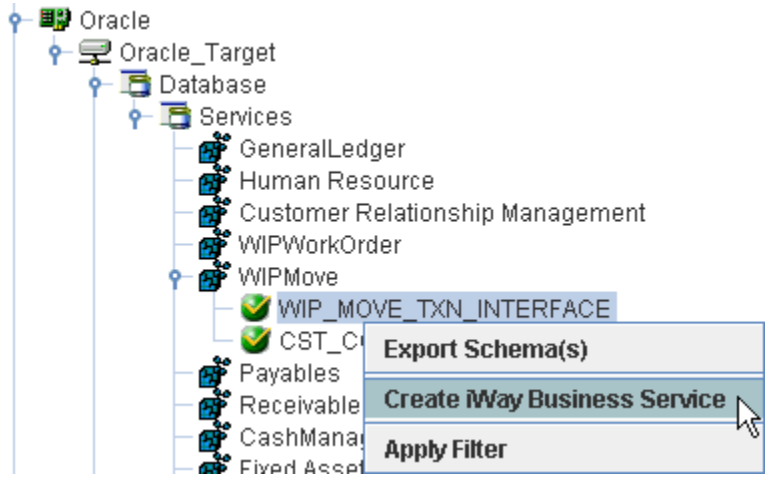
Ensure that the servlet iBSP is properly configured. For more information on installing and deploying iWay components, see the *iWay Installation and Configuration* manual.

Procedure: How to Create and Test a Web Service for an Interface Table

The following example shows how to create a web service for an Oracle E-Business Suite interface table.

1. Connect to an Oracle E-Business Suite target, as described in *Connecting to Oracle E-Business Suite* on page 53.
2. In the left pane, expand the target node.

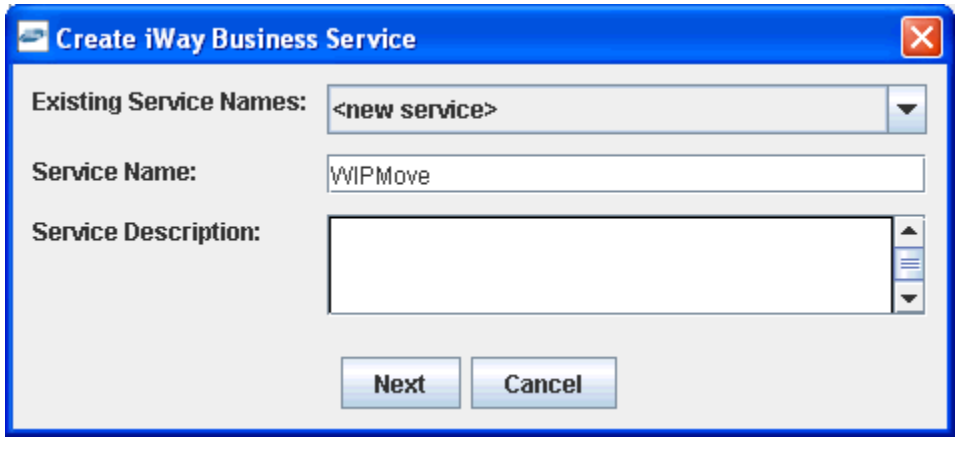
3. Locate and select an interface table, as described in *Browsing Interface Table Metadata* on page 67.



Note: The user ID for this target connection must have read access to the selected table, otherwise you cannot create a business service for it.

4. Right-click the interface table and select *Create iWay Business Service*.

   The Create iWay Business Service dialog box opens, as shown in the following image.



5. Perform the following steps:

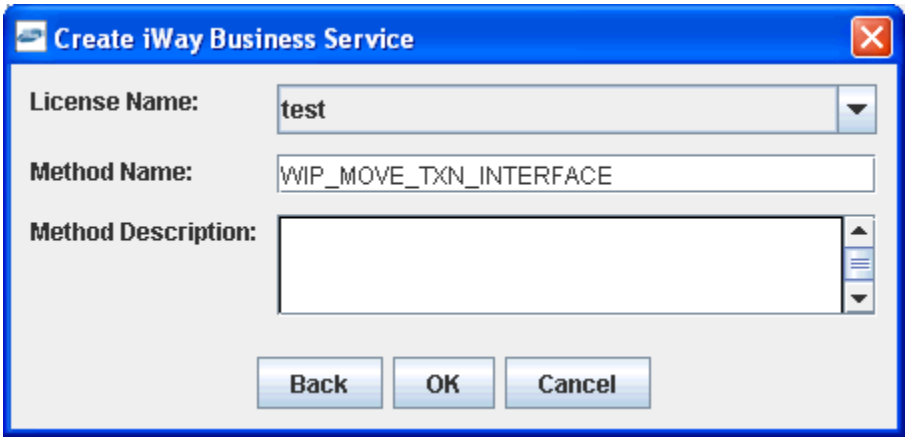
   a. From the Existing Service Names drop-down list, select whether you want to create a new service name or use an existing service name.

   b. In the Service Name field, type a descriptive name for the iWay Business Service.

c. In the Service Description field, type a brief description of the service (optional).

6. Click *Next*.

A second Create iWay Business Service dialog box opens and prompts you for additional information, as shown in the following image.



7. Perform the following steps:

a. From the License Name drop-down list, select a license definition.

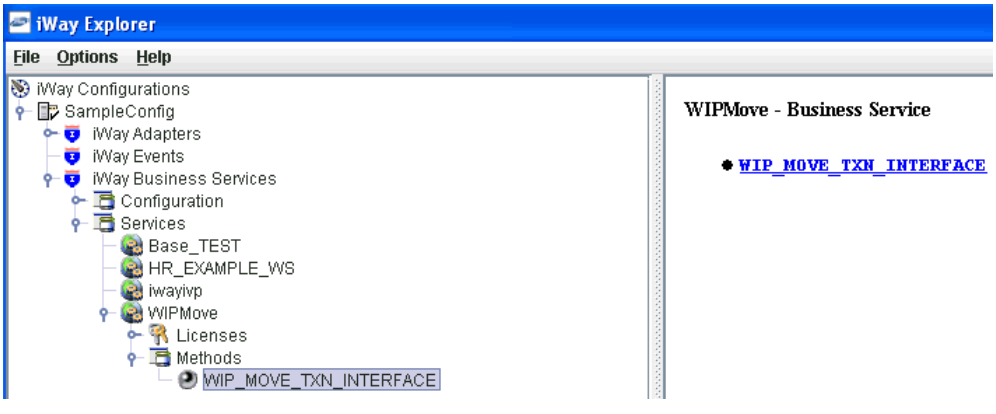
b. In the Method Name field, type a descriptive name for the method.

c. In the Method Description field, type a brief description of the method (optional).

8. Click *OK*.

The iWay Business Services node expands in the left pane. The new iWay Business Service appears under the Services node.

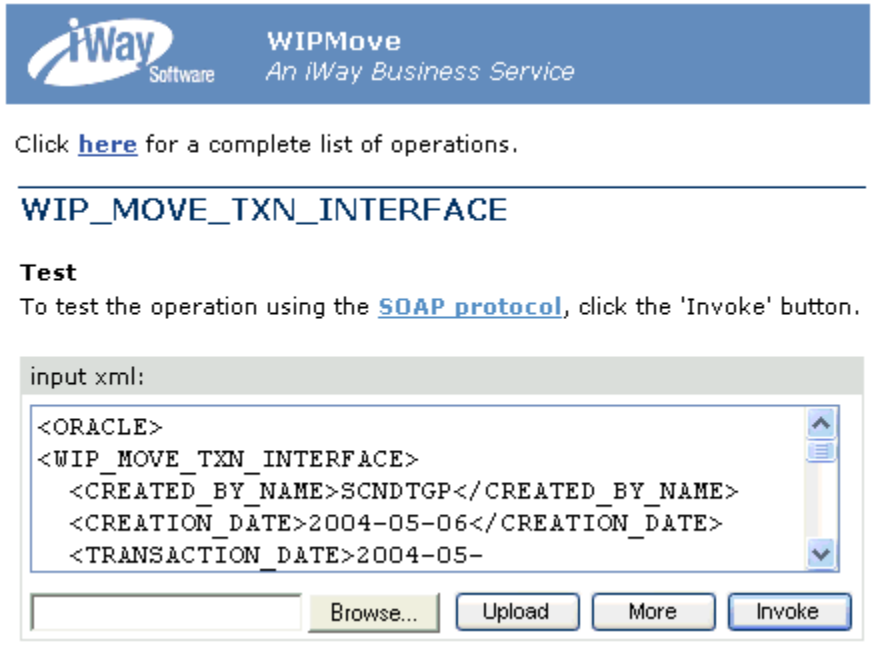
The following image shows an iWay Business Service called WIPMove and, under that service, a method called WIP_MOVE_TXN_INTERFACE.



The right pane displays the name of the expanded iWay Business Service and provides a hyperlink to the selected method, for example, WIP_MOVE_TXN_INTERFACE.

9. Click the *WIP_MOVE_TXN_INTERFACE* hyperlink in the right pane.

   An iWay Business Service test pane opens in your web browser, as shown in the following image.



10. Enter a sample XML document in the input xml field that will query the service.

    For a sample XML input document, see *Sample Input XML* on page 78.

11. Click *Invoke*.

    The test response appears in the web browser as shown in the following image.

Reference: Sample Input XML

The following section provides a sample input XML document that can be used to test an iWay Business Service using iWay Explorer.

```
<ORACLE>
<WIP_MOVE_TXN_INTERFACE>
  <CREATED_BY_NAME>SCNDTGP</CREATED_BY_NAME>
  <CREATION_DATE>2004-05-06</CREATION_DATE>
  <TRANSACTION_DATE>2004-05-06</TRANSACTION_DATE>
  <TRANSACTION_QUANTITY>1</TRANSACTION_QUANTITY>
  <TRANSACTION_UOM>Ea</TRANSACTION_UOM>
  <FM_OPERATION_SEQ_NUM>10</FM_OPERATION_SEQ_NUM>
  <TO_OPERATION_SEQ_NUM>20</TO_OPERATION_SEQ_NUM>
  <FM_INTRAOPERATION_STEP_TYPE>1</FM_INTRAOPERATION_STEP_TYPE>
  <TO_INTRAOPERATION_STEP_TYPE>2</TO_INTRAOPERATION_STEP_TYPE>
  <TRANSACTION_TYPE />
  <LAST_UPDATE_DATE>2004-05-06</LAST_UPDATE_DATE>
  <LAST_UPDATED_BY_NAME>SCNDTGP</LAST_UPDATED_BY_NAME>
  <WIP_ENTITY_NAME>49635</WIP_ENTITY_NAME>
  <PROCESS_PHASE>1</PROCESS_PHASE>
  <PROCESS_STATUS>1</PROCESS_STATUS>
  <PRIMARY_ITEM_ID>133</PRIMARY_ITEM_ID>
  <ORGANIZATION_CODE>M1</ORGANIZATION_CODE>
  </WIP_MOVE_TXN_INTERFACE>
  </ORACLE>
```

Running a Submit Request

iWay Application Adapter for Oracle E-Business Suite supports running what was formerly known as the Oracle Applications concurrent program function, using the adapter concurrent programs agent. Currently, this function is called the Submit Request.

The Submit Request submits a request that runs a concurrent program. Concurrent programs perform several functions, including moving data. For example, if you wish to move data to a base table:

1. First, write the data to an interface table.

   Each interface table has a Submit Request associated with it.

2. When the data is ready, run a Submit Request.

   The Submit Request moves the data from the interface table to the base table.

The Submit Request can call a concurrent program synchronously or asynchronously. It submits the concurrent program to a concurrent manager that controls background processing in Oracle E-Business Suite.

Procedure: How to Run the Submit Request

To run the Submit Request:

1. Use iWay Explorer to generate a pair of request and response schemas for the Submit Request, as described in *Generating a Schema for the Submit Request* on page 81.

   You do this only once for a Submit Request. This covers all Submit Requests to be run from a given Oracle E-Business Suite connection.

2. Add a service for the Submit Request, as described in *Generating a Web Service for the Submit Request* on page 81.

   You do this only once for a Submit Request. This covers all Submit Requests to be run from one Oracle E-Business Suite connection.

3. Create and test the request document, as described in *Creating a Request Document* on page 85.

Installing the Submit Request and Running the Concurrent.ora Script

To install the Submit Request, you must perform the following additional steps when installing the adapter:

1. Extract the Concurrent.ora script from the ibspsql.zip archive, which is located in the following directory:

   ```
   <iway_home>\etc\setup
   ```

2. Run the Concurrent.ora script against the Oracle database server using SQL*PLUS.

   Note: Contact Information Builders Customer Support Services to obtain the Concurrent.ora script.

   As part of the installation process, you must run the Concurrent.ora script. You can run it at any time before creating the service for the Submit Request. For information on how to run the Concurrent.ora script, see *How to Run the Concurrent.ora Script* on page 80.

3. Adjust the system path on the Oracle database server.
4. Verify the presence of the CONCSUB executable on the Oracle database server.

As part of the installation process, you must run the Concurrent.ora script. You can run it at any time before creating the service for the Submit Request. The computer on which you run the script must have:

- A TNSnames entry, in the Tnsnames.ora file, that points to the Oracle E-Business Suite database instance (on the Oracle server).

- ❑ SQL*Plus installed.

Procedure: How to Run the Concurrent.ora Script

To run the Concurrent.ora script:

1. Extract *Concurrent.ora* using WinZip or a similar extraction product.

   You can also use the iWay package facility to install the Oracle Applications package, which contains the script file. For more information, see the *iWay Service Manager User's Guide*.

2. After the Concurrent.ora script is extracted, move it to the computer where SQL*Plus resides, if it is not already located there.

3. Edit the Concurrent.ora script as follows:

   For Windows Platforms Only:

   In the `CREATE OR REPLACE AND RESOLVE JAVA SOURCE NAMED IwayConc...` section, the path to the cmd.exe file must be set correctly at `finalCommand[0]`. For example:

   ```
   finalCommand[0] = "C:\\WINDOWS\\system32\\cmd.exe";
   ```

   For Windows and UNIX Platforms:

   In the `CREATE OR REPLACE AND RESOLVE JAVA SOURCE NAMED IwayConc...` section, the path to the ConcSub.exe file must be appended before `concCommand` at `finalCommand[3]`. For example:

   ```
   finalCommand[3] = "D:\\oracle\visappl\\fnd\\11.5.0\\bin" + concCommand;
   ```

4. Open a command prompt or Telnet session.

5. In the directory in which Concurrent.ora resides, issue the following command using the APPS user ID:

   ```
   SQLPLUS APPS/password@database @Concurrent.ora
   ```

   where:

   password

   Is the password associated with the APPS user ID.

   database

   Is a value of the tnsnames entry (in the tnsnames.ora file) that points to the Oracle E-Business Suite database instance.

Procedure: How to Adjust the System Path and Verify CONCSUB

To adjust the system path and verify CONCSUB:

1. On the server where the Oracle E-Business Suite database instance resides, add the following to the system path:

   ```
   $FND_TOP\bin
   ```

   You can add it at any time before creating the service for the Submit Request.

2. Verify that the CONCSUB executable is present in $FND_TOP\bin.

   If CONCSUB is missing, contact your Oracle E-Business Suite administrator.

   $FND_TOP is the location of the Oracle E-Business Suite foundation directory. Its value is set in the vis.env (or prod.env) file located under the oraclehome\visappl directory.

Generating a Schema for the Submit Request

You must create a pair of request and response service schemas for the Submit Request. You need only do this once for each Oracle E-Business Suite connection from which you run these programs. This satisfies the requirement for all Submit Requests that are accessed using that connection.

Procedure: How to Generate a Schema for the Submit Request

To generate the request and response schemas:

1. If you have not already done so, connect to an Oracle E-Business Suite database instance.
2. In the left pane of iWay Explorer under the Oracle Applications target, expand the *Services* node and the *submit_request* node.
3. Select *submit_request*.
4. In the right pane, move the pointer over *Operations* and select *Generate Schema*.

   A table appears on the right.

5. To review a schema, click the ellipsis (...) in the Schemas column for the appropriate row.

You have finished generating the service schemas.

Generating a Web Service for the Submit Request

Generating a web service for the submit request is identical to configuring other Oracle E-Business Suite services, except that some of the configuration parameters differ.

The following table lists and describes the service parameters for the submit request.

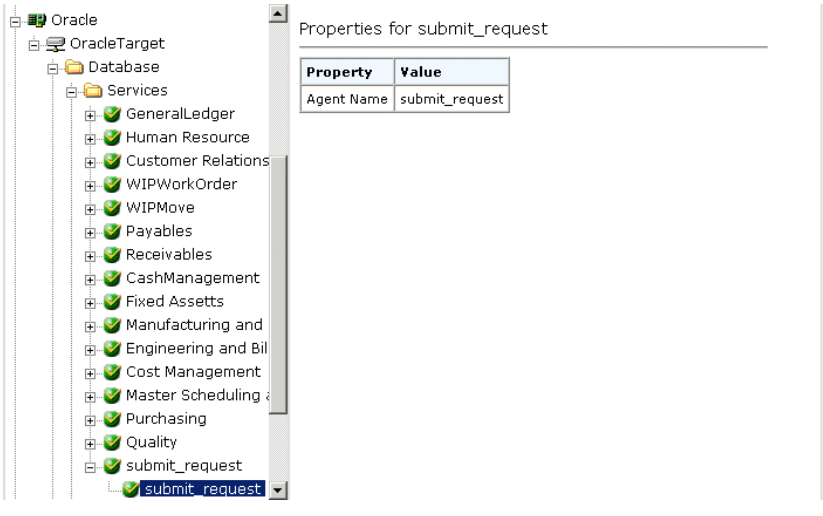
Parameter	Description
Select	Submit Request.
Userid	User ID for the Oracle E-Business Suite database.
Password	Password associated with the user ID.
Host	Name of the server where the Oracle E-Business Suite database instance resides.
Port	Port number on which the database is listening.
ServiceId	Unique name, comprised of the database name and domain name, that identifies the database.
TnsName	Logical name of the Oracle E-Business Suite database instance.

Procedure: How to Generate a Web Service for Submit Request

To create a web service:

1. In the left pane of iWay Explorer, expand the *submit_request* node.

   The following image shows the expanded submit_request service node on the left and a property table for the submit request service that opens on the right.

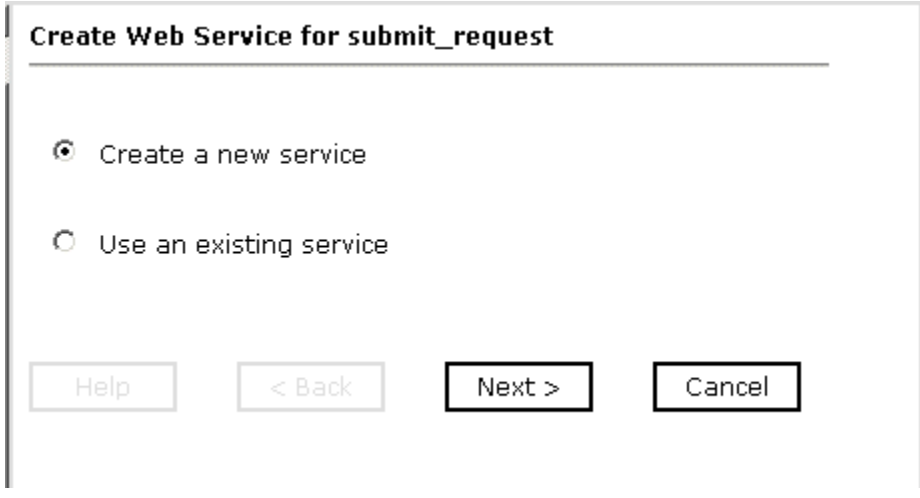


2. Select *submit_request*.

   The Operations menu becomes available on the right.

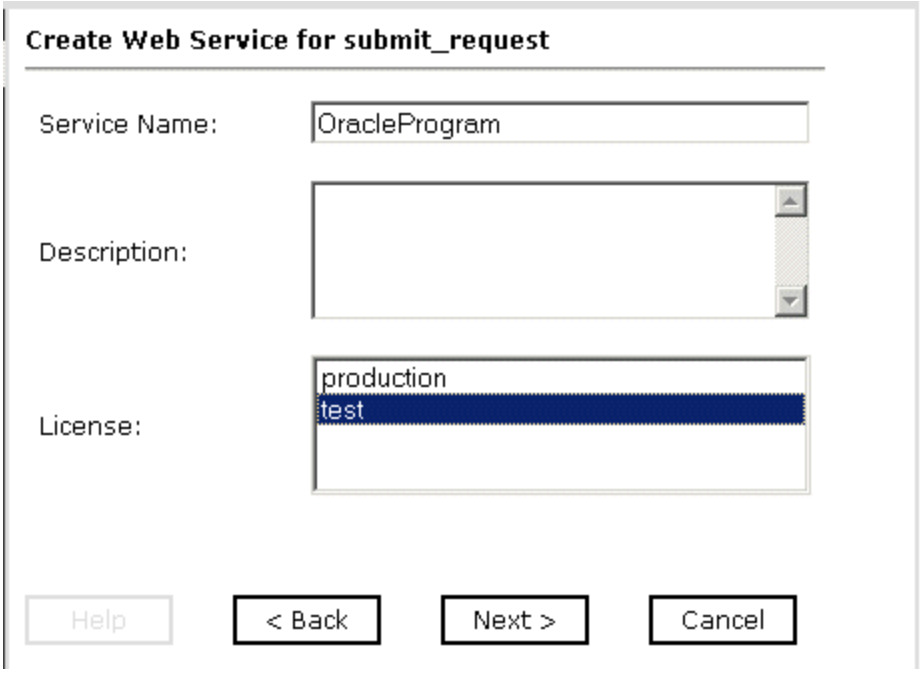
3. Move the pointer over *Operations* and select *Create iWay Business Services*.

   The Create Web Service for submit_request pane opens on the right and provides the option to create a new service or use an existing service, as shown in the following image.



4. Select the *Create a new service* option button.
5. Click *Next*.

The right pane prompts you for descriptive information about the service, as shown in the following image.



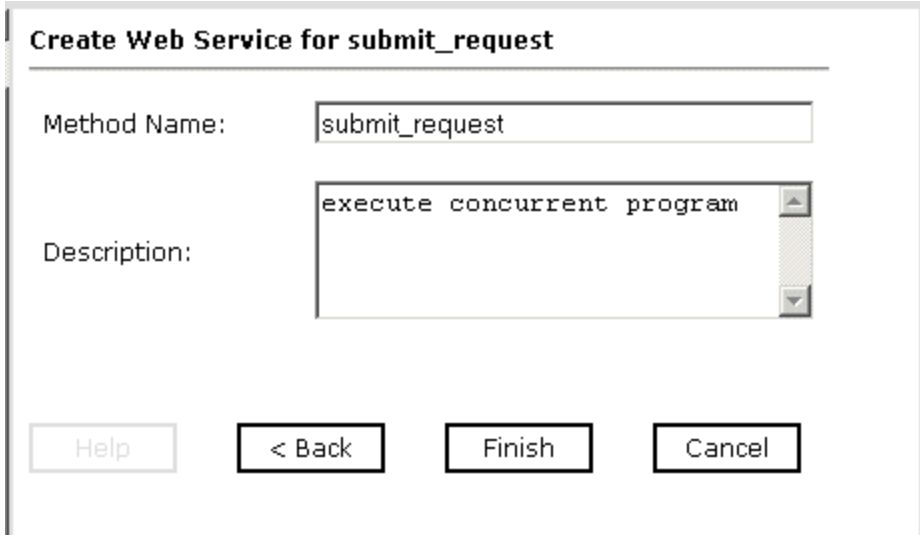
a. In the Service Name field, type a name for the web service.

b. In the Description field, type a brief description of the web service.

c. From the License list, select the license, for example, test.

6. Click *Next*.

The right pane then prompts you to identify a method for the service, as shown in the following image.



a. In the Method Name field, type a name for your method.

b. In the Description field, type a brief description of the method.

7. Click *Finish*.

Creating a Request Document

You can use an XML editor to create the service request document. For a sample submit request document, see *How to Test the Submit Request* on page 85.

After you create a web service and a request document, you can verify that you correctly configured the service for the Submit Request. After you create a service, the test pane appears on the right.

Procedure: How to Test the Submit Request

To test the Submit Request service:

1. Ensure the Test pane appears on the right after you create the service. If it does not appear:

   a. Click the *iWay Business Services* tab.

   b. Click the *Browse* button.

   c. Use the tree in the left pane to browse to your service and method.

2. Copy the XML input from the request document and paste it in the input xml field of the test pane. For a sample XML input document, see *Sample Input XML* on page 86.
3. To test the Submit Request, click *Invoke*.

   The response document appears on the right.

   The *submit_response_id* returned in the reply is the request id returned from the concsub command. This request id can be used through Oracle E-Business Suite to monitor the status of the concurrent manager program instance that was invoked.

You have completed the testing of the Submit Request service.

Reference: **Sample Input XML**

The following is a sample input XML document that can be used to test an iWay Business Service for an Oracle concurrent program:

```
<?xml version="1.0" encoding="UTF-8"?>
<ORACLE>
  <submit_request>
    <resp_appl_shrtnm>SYSADMIN</resp_appl_shrtnm>
    <responsibility>System Administrator</responsibility>
    <username>SYSADMIN</username>
    <wait>N</wait>
    <prog_appl_shrtnm>WIP</prog_appl_shrtnm>
    <program>WICTMS</program>
<parm>3993</parm>
<parm>0</parm>
<parm>1</parm>
  </submit_request>
</ORACLE>
```

Chapter 7

Integrating With Oracle APIs

This section describes how to browse metadata, generate XML schemas, and create iWay Business Services for Oracle APIs.

Although this section shows the Java Swing implementation of iWay Explorer, other implementations provide the same functionality by means of similar graphical user interfaces.

In this chapter:

Browsing Oracle API Metadata

Oracle E-Business Suite supports PL/SQL blocks, package. A package is a PL/SQL construct that allows related objects to be stored together. The adapter is able to expose all the packages and interact with its stored procedure element. In the iWay Application Adapter for Oracle E-Business Suite, packages are categorized by the schema that owns the object. Using iWay Explorer, you can search for specific packages and specific stored procedures under a package.

Note: If you do not see a package under a schema, you may need to create a public synonym. An Oracle DBA may use the following syntax:

```
create [or replace] [public] synonym schema.synonym_namefor
schema.object_name [@ dblink];
```

where:

`or replace`

Allows you to replace a synonym that already exists, without having to issue a DROP synonym command (optional).

`public`

Indicates that the synonym is a public synonym and is accessible to all users. You must have the appropriate privileges to the object to use the synonym (optional).

schema

Is the name of the appropriate schema. If this phrase is omitted, Oracle assumes that you are referring to your own schema.

synonym_name

Is the name of the public synonym you are creating.

object_name

Is the name of the object for which you are creating the synonym.

For example:

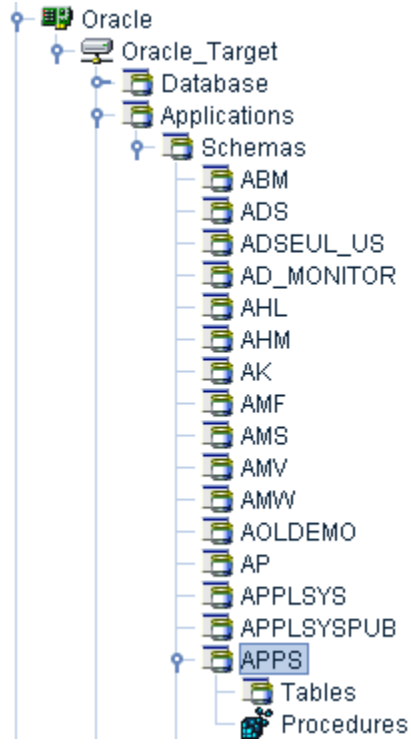
```
create public synonym suppliers
for app.suppliers;
```

Procedure: How to Search for Packages

To locate a package and view the Oracle E-Business Suite stored procedures it contains:

1. Connect to an Oracle E-Business Suite target, as described in *Connecting to Oracle E-Business Suite* on page 53.
2. In the left pane, expand the target node.

3. Expand *Applications*, *Schemas*, and then *APPS*, as shown in the following image:



4. Select *Procedures*, and then *Packages*.

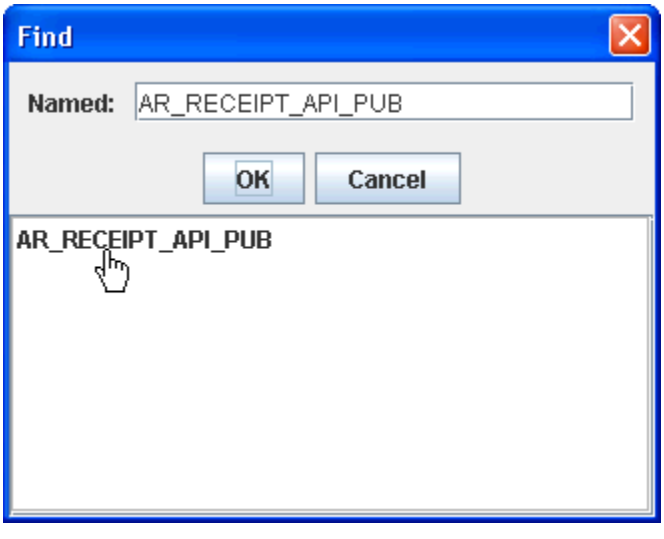
5. Right-click *Packages* and select *Find*.

The Find dialog box opens.

Note: Packages are collections of stored procedures. Due to the large number of packages and stored procedures available, a search capability is provided. You can search for a particular package by typing the full package name into the Named field, as shown in the preceding image. Alternatively, you can search for package names beginning with a certain string, for example, typing *FND%* generates a list of all package names starting with the string FND. This search capability is not case sensitive.

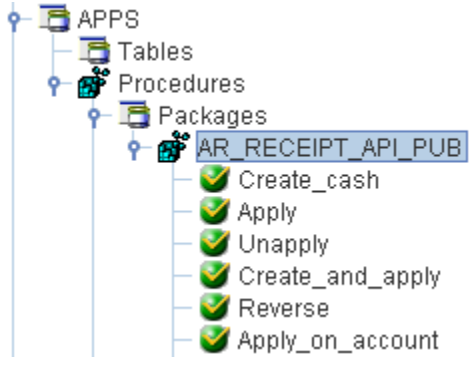
6. Click *OK*.

   The Find dialog box expands and lists the search results, as shown in the following image:



7. Click the package name.

The AR_RECEIPT_API_PUB package is listed under Packages in the left pane.



8. Expand the AR_RECEIPT_API_PUB node.

   A list of the stored procedures contained in this package is displayed in the left pane.

   You can now generate schemas and create iWay Business Services for packages.

Generating XML Schemas for Oracle APIs

When you deploy the iWay Application Adapter for Oracle E-Business Suite in the iWay Business Services Provider (iBSP) environment, you can generate schemas that define a service request document and the corresponding response document.

If you plan to deploy the iWay Application Adapter for Oracle E-Business Suite in a business services environment, you are not required to generate a schema.

Note: Schemas for Oracle APIs with non-complex data types are RPC style.

Schema Location

By default, iWay Explorer stores the schemas it creates in subdirectories under the iWay home directory of the machine where it is installed. However, you can also export these schemas using iWay Explorer to any location you specify on your file system.

The exact location of the schemas differs depending on whether you deploy iWay Explorer with an iBSP configuration.

When the adapter is used with an iBSP configuration, iWay Explorer stores the schemas in a subdirectory of the iWay installation directory, for example,

iWayHome\config\base\wsdl\schemas\service\Oracle*Oracle_Target*

where:

Oracle_Target

Is the name of the connection (target) to the Oracle Applications system that you defined using iWay Explorer. Under this directory, iWay Explorer creates subdirectories containing schemas.

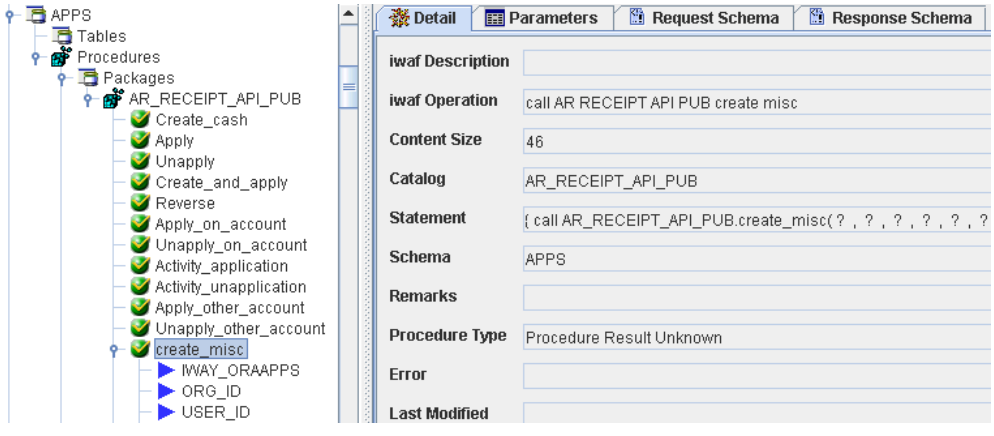
Procedure: How to Generate a Schema for a Stored Procedure

To generate a schema for an Oracle E-Business Suite stored procedure using iWay Explorer:

1. Connect to an Oracle E-Business Suite target, as described in *Connecting to Oracle E-Business Suite* on page 53.
2. Locate a stored procedure, as described in *Browsing Oracle API Metadata* on page 87.
3. Click a stored procedure, for example, *create_misc*.

   XML request and response schemas for the stored procedure are automatically created and displayed.

   The Detail, Parameters, Request Schema, and Response Schema tabs appear on the right.

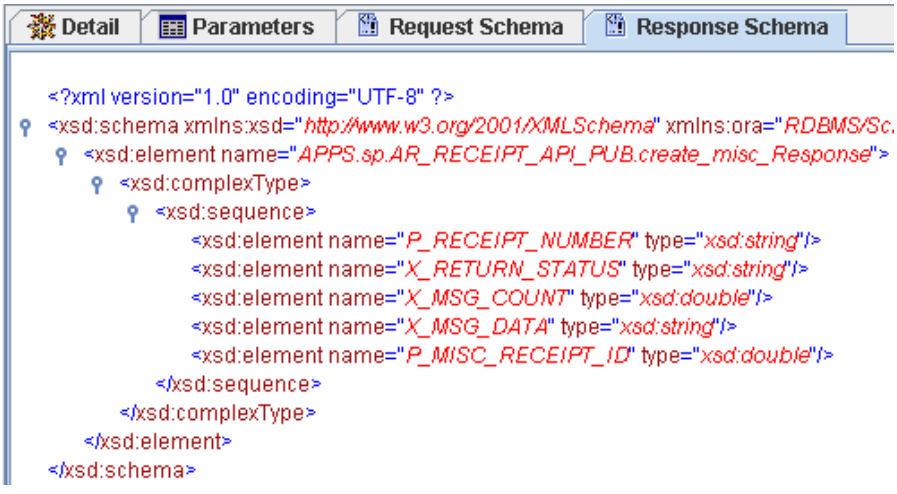


4. Click the *Request Schema* tab in the right pane.

The XML request schema appears in the right pane, as shown in the following image:

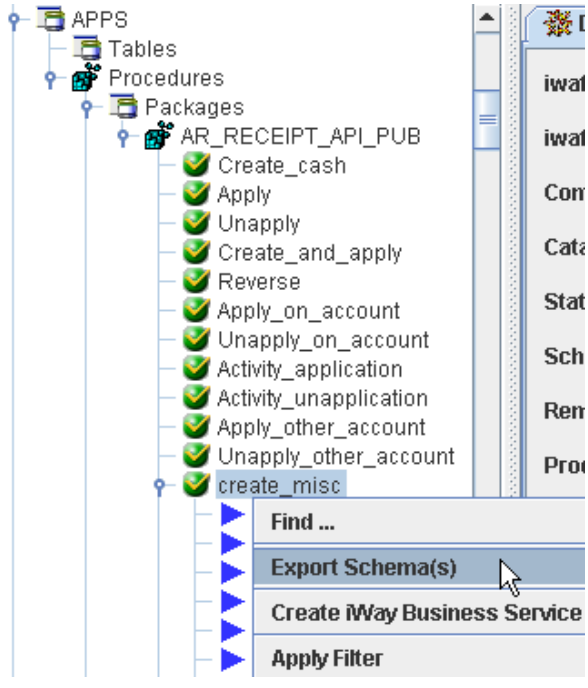
5. Click the *Response Schema* tab in the right pane.

The XML response schema appears in the right pane, as shown in the following image:

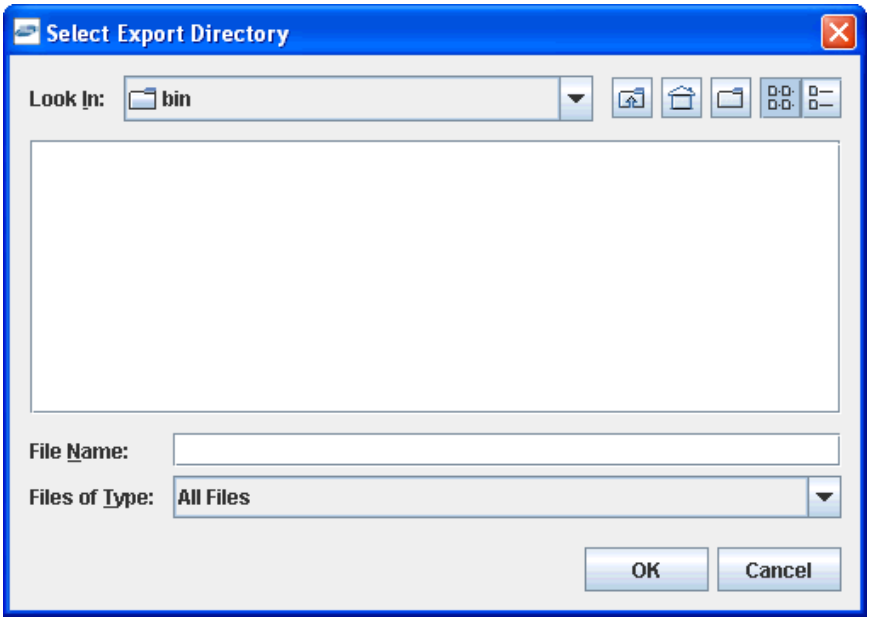


Note: When creating XML schemas for stored procedures, in certain cases there may be no output element in the response schema. This happens when in the code of a stored procedure there are inputs but no outputs. In cases where there are no outputs, there will be no output element in the response schema. For detailed information on individual stored procedures, see your Oracle E-Business Suite documentation.

6. To export XML schemas, right-click the table in the left pane, and select *Export Schemas*, as shown in the following image:



The Select Export Directory dialog box opens, as shown in the following image:

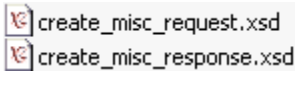


7. Navigate to a directory on your file system where you want to export the XML schemas.

   The file path displays in the File Name field.

8. Click *OK*.

   The XML request and response schemas are now exported to your local file system.

Creating iWay Business Services for Oracle APIs

You can generate an iWay Business Service (also known as a web service) for Oracle E-Business Suite APIs. To generate an iWay Business Service, you must deploy the iWay Application Adapter for Oracle E-Business Suite in a business services environment using the iWay Business Services Provider (iBSP). iBSP exposes functionality as web services and serves as a gateway to heterogeneous back-end applications and databases.

A web service is a self-contained, modularized function that can be published and accessed across a network using open standards. It is the implementation of an interface by a component and is an executable entity. For the caller or sender, a web service can be considered a "black box" that may require input and delivers a result. Web services can be integrated within an enterprise as well as across enterprises on any communication technology stack, whether asynchronous or synchronous, in any format.

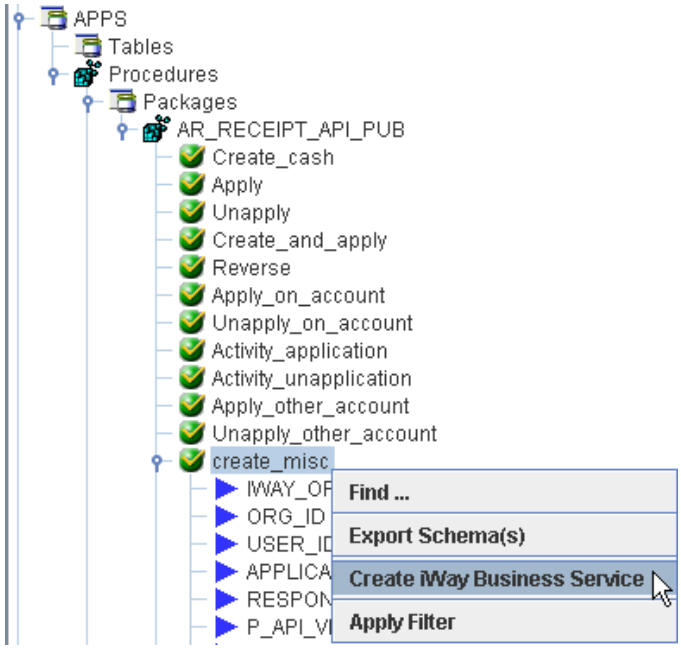
You can make a web service available to other services within a host server by generating WSDL (Web Services Description Language) from the web service.

Ensure that the servlet iBSP is properly configured. For more information on installing and deploying iWay components, see the *iWay Installation and Configuration* manual.

Procedure: How to Create and Test a Web Service for a Stored Procedure

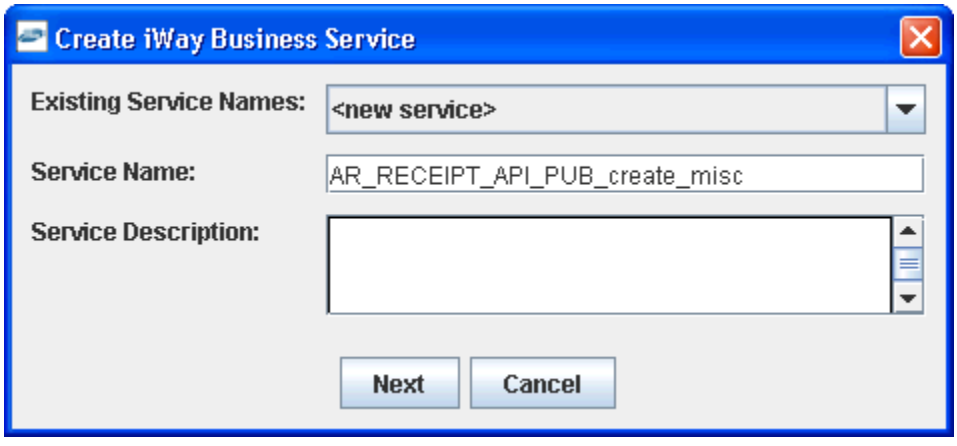
To create a web service for an Oracle E-Business Suite stored procedure:

1. Connect to an Oracle E-Business Suite target, as described in *Connecting to Oracle E-Business Suite* on page 53.
2. Locate and select a stored procedure under AR_RECEIPT_API_PUB, as described in *How to Search for Packages* on page 88.



3. Right-click the stored procedure, for example, create_misc, and select *Create iWay Business Service*.

The Create iWay Business Service dialog box opens, as shown in the following image.



4. Perform the following steps:

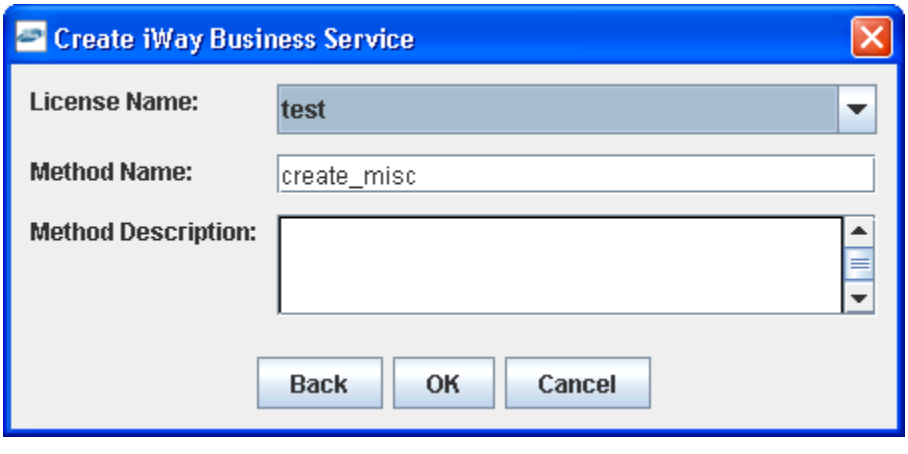
   a. From the Existing Service Names drop-down list, select whether you want to create a new service name or use an existing service name.

   b. In the Service Name field, type a descriptive name for the iWay Business Service.

   c. In the Service Description field, type a brief description of the service (optional).

5. Click *Next*.

   A second Create iWay Business Service dialog box opens and prompts you for additional information, as shown in the following image.



6. Perform the following steps:

   a. From the License Name drop-down list, select a license definition.

b. In the Method Name field, type a descriptive name for the method.

c. In the Method Description field, type a brief description of the method (optional).

7. Click *OK*.

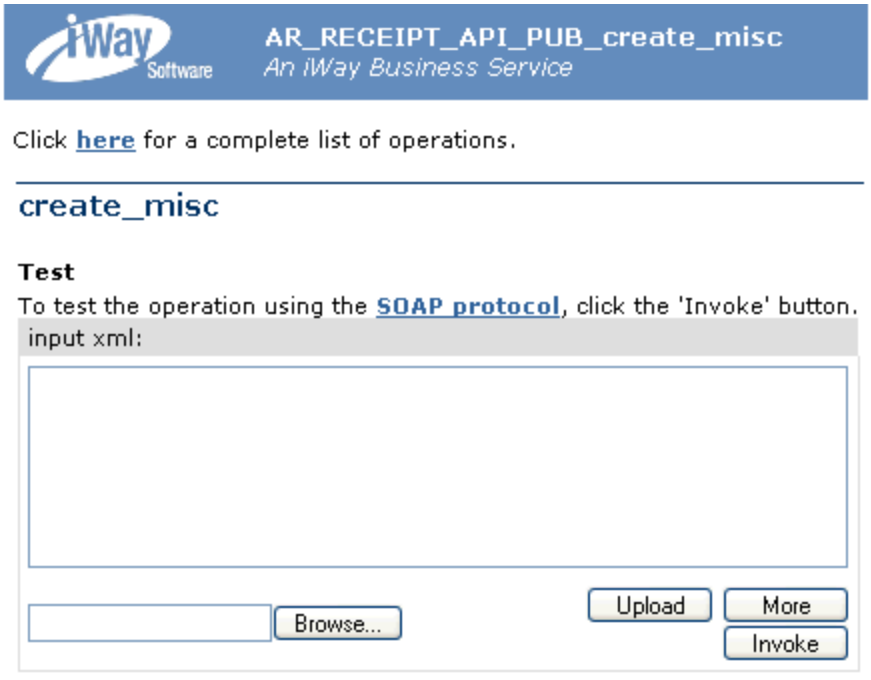
   The iWay Business Services node expands in the left pane. The new iWay Business Service appears under the Services node.

   The following image shows an iWay Business Service called AR_RECEIPT_API_PUB_create_misc and, under that service, a method called create_misc.

   The right pane displays the name of the expanded iWay Business Service and provides a hyperlink to the selected method, for example, create_misc.

8. Click the *create_misc* hyperlink in the right pane.

An iWay Business Service test pane opens in your web browser, as shown in the following image.



9. Enter a sample XML document in the input xml field that will query the service.

   For more information, see *Sample Input XML* on page 100.

10. Click *Invoke*.

    The test response appears in the web browser.

Reference: Sample Input XML

The following section provides a sample input XML document that can be used to test an iWay Business Service using iWay Explorer.

```
<APPS.sp.AR_RECEIPT_API_PUB.create_misc_Request>
<IWAY_ORAAPPS>Y</IWAY_ORAAPPS>
                <ORG_ID>204</ORG_ID>
<IWAY_ORAAPPS>Y</IWAY_ORAAPPS>
                <USER_ID>1</USER_ID>
                <APPLICATION_ID>222</APPLICATION_ID>
                <RESPONSIBILITY_ID>20678</RESPONSIBILITY_ID>
                <P_API_VERSION>1.0</P_API_VERSION>
                <P_INIT_MSG_LIST>T</P_INIT_MSG_LIST>
                <P_CURRENCY_CODE>USD</P_CURRENCY_CODE>

                <P_AMOUNT>477</P_AMOUNT>
                <P_RECEIPT_NUMBER>18257</P_RECEIPT_NUMBER>
                <P_RECEIPT_DATE>2005-12-12</P_RECEIPT_DATE>
                <P_GL_DATE>2005-12-12</P_GL_DATE>
                <P_RECEIVABLES_TRX_ID>1000</P_RECEIVABLES_TRX_ID>


                <P_RECEIPT_METHOD_ID>1002</P_RECEIPT_METHOD_ID>

                <P_ATTRIBUTE_RECORD recordType="">

                    <ATTRIBUTE1>Chatura</ATTRIBUTE1>

                </P_ATTRIBUTE_RECORD>
                <P_GLOBAL_ATTRIBUTE_RECORD recordType="">

                    <GLOBAL_ATTRIBUTE1>iway</GLOBAL_ATTRIBUTE1>

                </P_GLOBAL_ATTRIBUTE_RECORD>
                <P_COMMENTS>This is a test</P_COMMENTS>

            </APPS.sp.AR_RECEIPT_API_PUB.create_misc_Request>
```

Chapter 8

Integrating With Oracle Base Tables

This section describes how to browse metadata, generate XML schemas, and create iWay Business Services for Oracle base tables.

Although this section shows the Java Swing implementation of iWay Explorer, other implementations provide the same functionality by means of similar graphical user interfaces.

In this chapter:

Browsing Base Table Metadata

iWay Application Adapter for Oracle E-Business Suite now supports direct interaction with Oracle base tables and views. This integration method is most appropriate when you need to query data located in base tables and views.

The following functions are supported:

- SELECT
- CURSOR
- COUNT
- AVERAGE
- MAX
- MIN
- SUM
- GET
- INSERT

- UPDATE
- DELETE
- UPSERT

Important: Oracle recommends that users do not interact directly with application tables. The use of INSERT, DELETE, or UPDATE to modify data in application tables may jeopardize database referential integrity.

Procedure: How to Search for Oracle Base Tables and Views

To locate a specific Oracle base table using iWay Explorer:

1. Connect to an Oracle E-Business Suite target, as described in *Connecting to Oracle E-Business Suite* on page 53.
2. In the left pane, expand the target node.
3. Expand *Applications*, *Schemas*, and then *HR*.

   Note: The schemas under Applications are actual table owners.

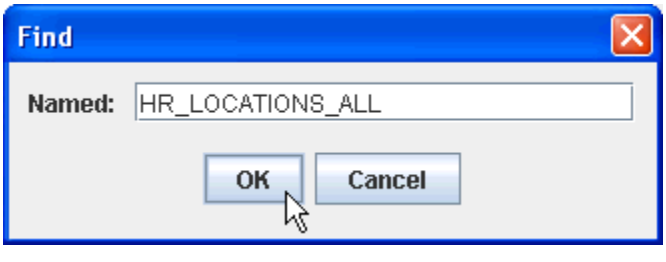
   

4. Right-click *Tables* and select *Find*.

   Note: The Tables node provides the capability to interact with base tables and views.

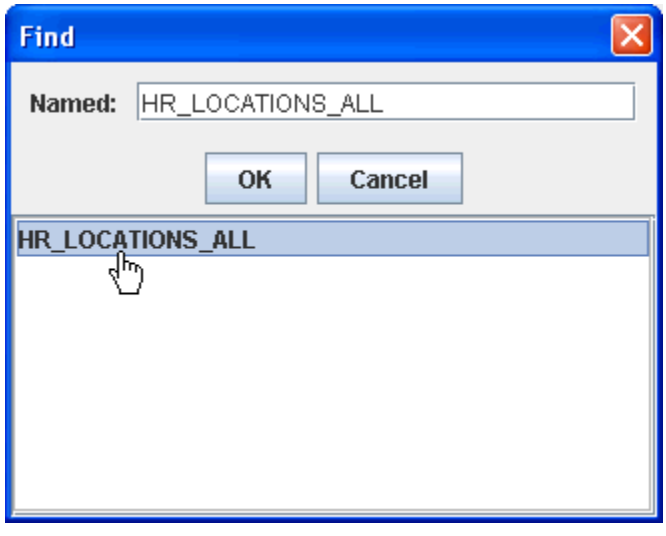
   The Find dialog box opens.

Note: Due to the large number of tables available, a search capability is provided. You can search for a particular table by typing the full table name into the Search string field, as shown in the preceding image. Alternatively, you can search for table names beginning with a certain string, for example, typing *HR_LOC%* generates a list of all tables starting with the string HR_LOC. This search capability is not case sensitive.

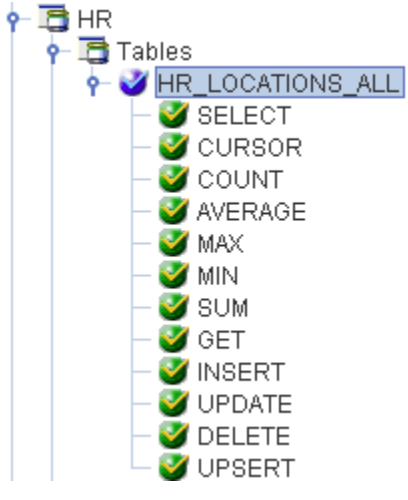
5. Click *OK*.

   The Find dialog box expands and lists the search results, as shown in the following image:



6. Click the table name.

   The HR_LOCATIONS_ALL table is listed under Tables in the left pane.



7. Expand the HR_LOCATIONS_ALL node.

   A list of functions that can be performed on this table is displayed in the left pane.

   Important: Oracle recommends that users do not interact directly with application tables. The use of INSERT, DELETE, or UPDATE to modify data in application tables may jeopardize database referential integrity.

   You can now generate schemas and create iWay Business Services.

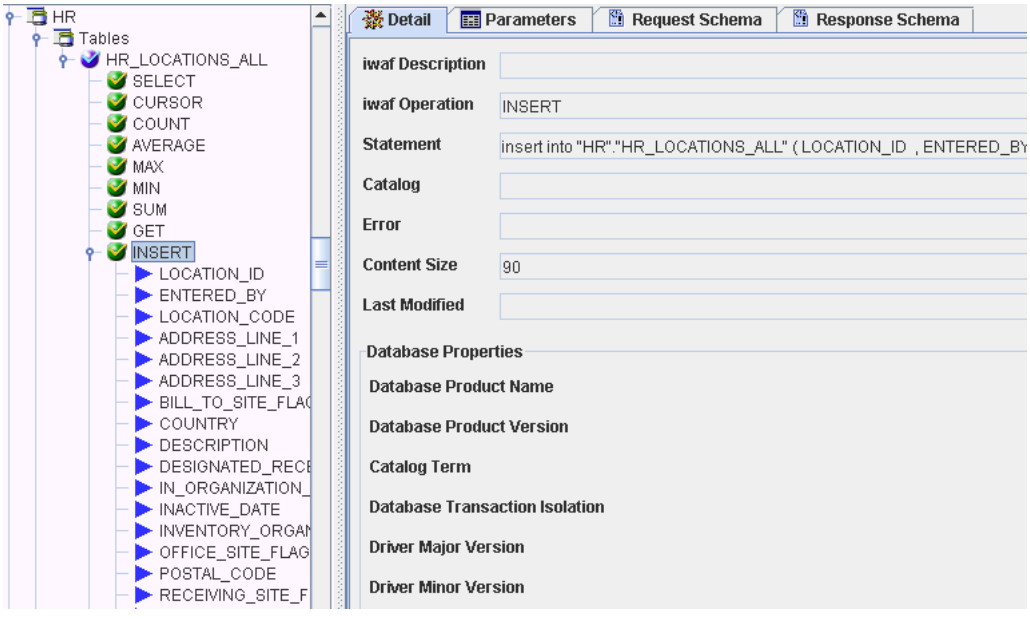
Procedure: How to Use the Additional Table Functions

The additional table functions provide a standard way to store, update, and retrieve data from any database. The GET, INSERT, UPDATE, DELETE, and UPSERT functions operate on a single row at a time. The COUNT, AVERAGE, MIN, MAX, and SUM functions operate on a single table at a time.

To use the additional table functions:

1. After connecting to a target, expand *Applications* and then *Schemas*.
2. Search for a specific table.
3. Select a function in the left pane.

All the table functions can be used to create iWay Business Services and to export schemas to create instance XML request documents. You can review the SQL statement in the right pane when you select one of the functions. The following image shows the Insert function selected and the SQL statement displayed in the right pane.



The following table lists and describes the available table functions.

Function	Description
GET	Retrieves one row at a time.
INSERT	Inserts one row into a table.
UPDATE	Updates non-primary keys and operates on a single row at a time. The primary keys sent in the request are used to populate the where clause in the SQL statement. Updating primary keys is logically equivalent to deleting the row by primary key and then inserting a new row with a new ID with the same non-primary key values. This logic fits the DAO, JDO, or EJB frameworks while simply updating the primary keys does not.

Function	Description
DELETE	Deletes one row at a time. The parameters are the primary keys.
UPSERT	Combines the INSERT and UPDATE functions. This function checks if a row already exists. If it does not exist, the request is forwarded to the INSERT function; this is determined by performing a count query with the table's primary keys as part of the where clause, as shown in the properties in the right pane.
COUNT	The COUNT table function is used to return the number of entries in a table. The query does not accept any input parameters.
AVERAGE	This query executes an avg() function on a table and accepts an input parameter representing the column name. The JDBC API is not used to set the question mark in the statement. This process recreates the statement string replacing the question mark with the value in the input parameter in the request. This is performed before the statement is prepared.
MAX	This query executes a max() function on a table and accepts an input parameter representing the column name.
MIN	This query executes a min() function on a table and accepts an input parameter representing the column name.
SUM	This query executes a sum() function on a table and takes in an input parameter representing the column name.

Tables with no primary keys have only CURSOR and INSERT functions available. Tables that are actually views will have only the CURSOR function available.

4. In the left pane, select the function you want to use.
5. In the right pane, move the mouse pointer over *Operations* and select either *Create iWay Business Service* or *Generate Schema*.

Generating XML Schemas for Base Tables

When you deploy the iWay Application Adapter for Oracle E-Business Suite in the iWay Business Services Provider (iBSP) environment, you can generate schemas that define a service request document and the corresponding response document.

If you plan to deploy the iWay Application Adapter for Oracle E-Business Suite in a business services environment, you are not required to generate a schema.

Note: Schemas for Oracle base tables are RPC style.

Schema Location

By default, iWay Explorer stores the schemas it creates in subdirectories under the iWay home directory of the machine where it is installed. However, you can also export these schemas using iWay Explorer to any location you specify on your file system.

The exact location of the schemas differs depending on whether you deploy iWay Explorer with an iBSP configuration.

When the adapter is used with an iBSP configuration, iWay Explorer stores the schemas in a subdirectory of the iWay installation directory, for example,

iWayHome\config\base\wsdl\schemas\service\Oracle*Oracle_Target*

where:

Oracle_Target

Is the name of the connection (target) to the Oracle Applications system that you defined using iWay Explorer. Under this directory, iWay Explorer creates subdirectories containing schemas.

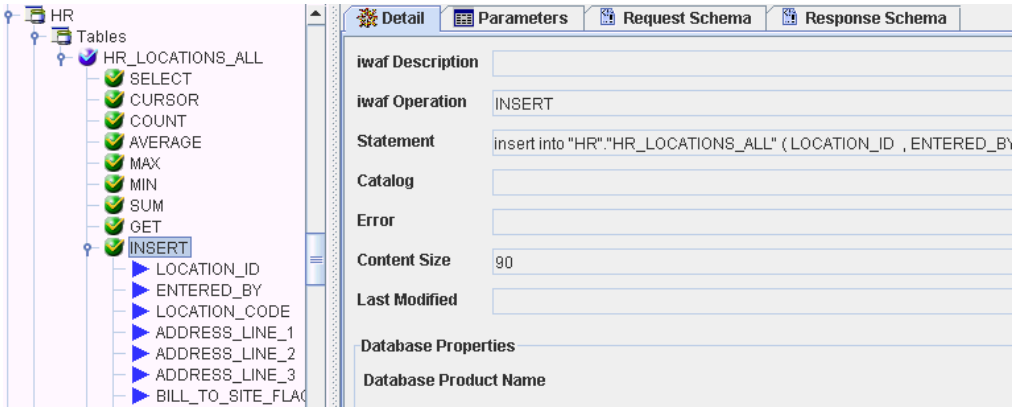
Procedure: How to Generate a Schema for a Base Table or View

To create a pair of request and response service schemas for interaction with an Oracle base table using iWay Explorer:

1. Connect to an Oracle E-Business Suite target, as described in *Connecting to Oracle E-Business Suite* on page 53.
2. Locate an Oracle base table, for example, HR_LOCATIONS_ALL, as described in *Browsing Base Table Metadata* on page 103.
3. Click *INSERT*.

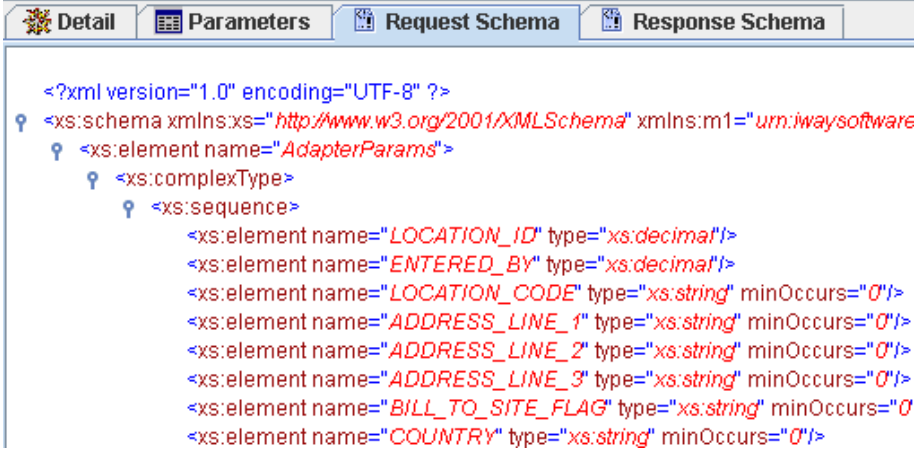
   Important: Oracle recommends that users do not interact directly with application tables. The use of INSERT, DELETE, or UPDATE to modify data in application tables may jeopardize database referential integrity.

XML request and response schemas for the base table are automatically created and displayed.



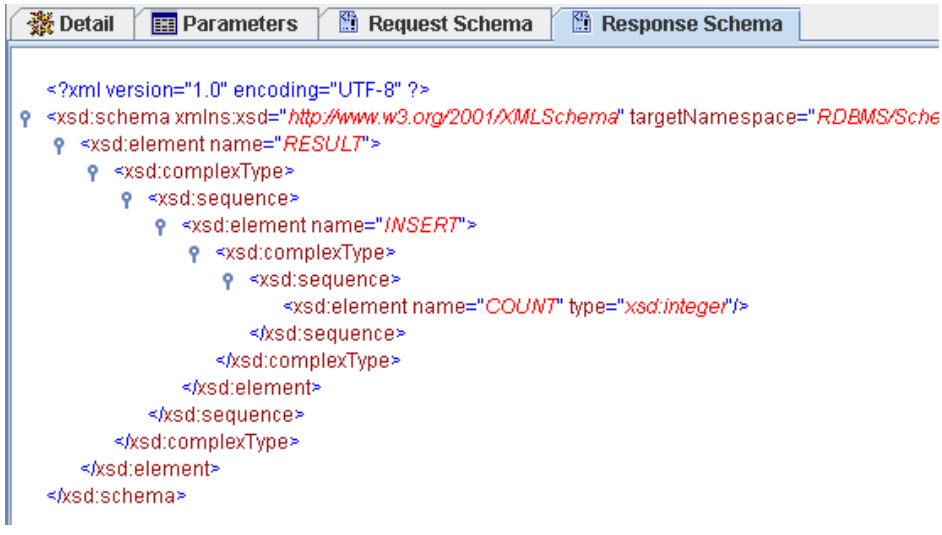
4. Click the *Request Schema* tab in the right pane.

   The XML request schema appears in the right pane, as shown in the following image:

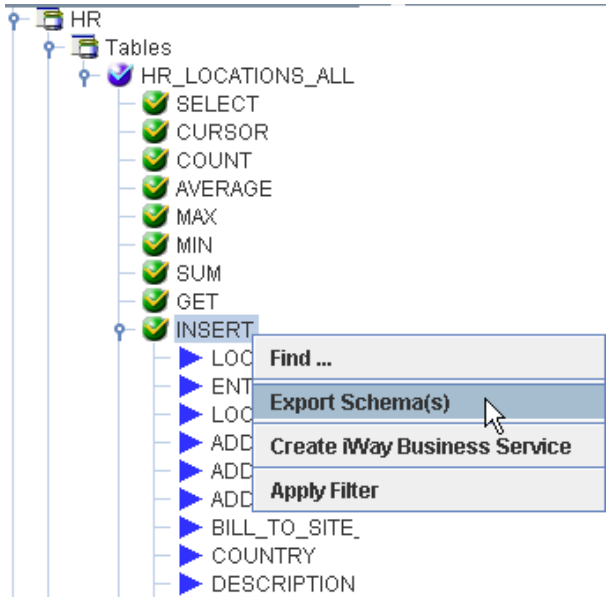


5. Click the *Response Schema* tab in the right pane.

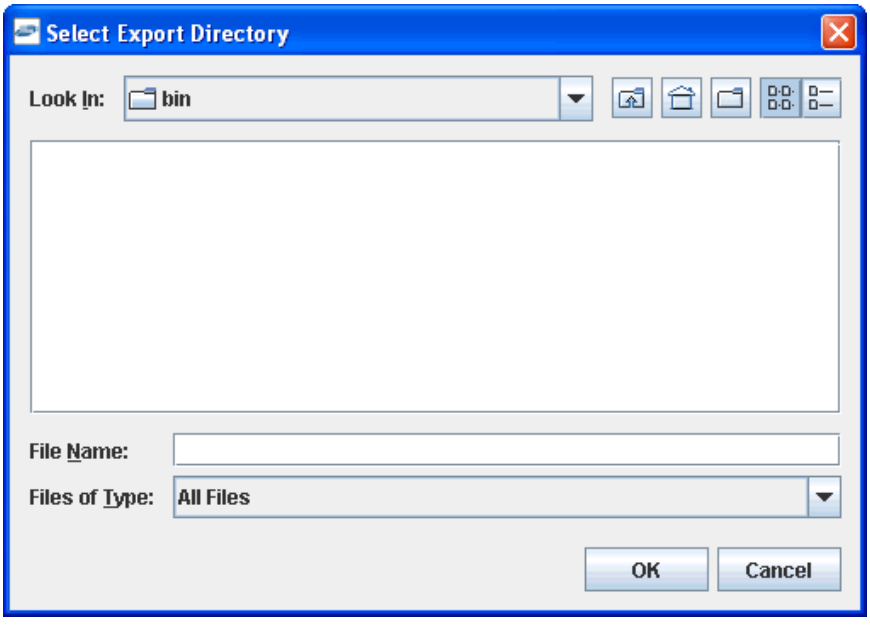
The XML response schema appears in the right pane, as shown in the following image:



6. To export XML schemas, right-click the table in the left pane, and select *Export Schemas*, as shown in the following image:



The Select Export Directory dialog box opens, as shown in the following image:



7. Navigate to a directory on your file system where you want to export the XML schemas.

   The file path displays in the File Name field.

8. Click *OK*.

   The XML request and response schemas are now exported to your local file system.

   SELECT_request.xsd
   SELECT_response.xsd

Creating iWay Business Services for Base Tables

You can generate an iWay Business Service (also known as a web service) for an Oracle E-Business Suite base table. To generate an iWay Business Service, you must deploy the iWay Application Adapter for Oracle E-Business Suite in a business services environment using the iWay Business Services Provider (iBSP). iBSP exposes functionality as web services and serves as a gateway to heterogeneous back-end applications and databases.

A web service is a self-contained, modularized function that can be published and accessed across a network using open standards. It is the implementation of an interface by a component and is an executable entity. For the caller or sender, a web service can be considered a "black box" that may require input and delivers a result. Web services can be integrated within an enterprise as well as across enterprises on any communication technology stack, whether asynchronous or synchronous, in any format.

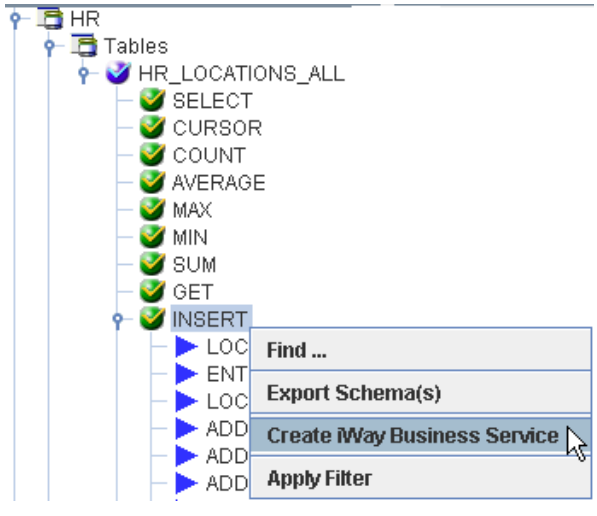
You can make a web service available to other services within a host server by generating WSDL (Web Services Description Language) from the web service.

Ensure that the servlet iBSP is properly configured. For more information on installing and deploying iWay components, see the *iWay Installation and Configuration* manual.

Procedure: How to Create and Test a Web Service for a Base Table

To create a web service for the Oracle E-Business Suite base table HR_LOCATIONS_ALL:

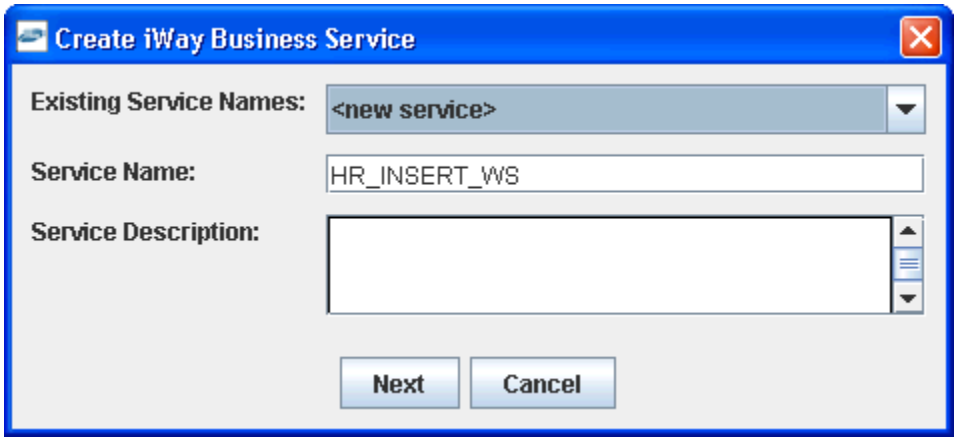
1. Connect to an Oracle E-Business Suite target, as described in *Connecting to Oracle E-Business Suite* on page 53.
2. Locate and expand the HR_LOCATIONS_ALL table node, as described in *How to Search for Oracle Base Tables and Views* on page 104.



Important: Oracle recommends that users do not interact directly with application tables. The use of INSERT, DELETE, or UPDATE to modify data in application tables may jeopardize database referential integrity.

3. Right-click a function for the base table, for example, INSERT, and select *Create iWay Business Service*.

The Create iWay Business Service dialog box opens, as shown in the following image.



4. Perform the following steps:

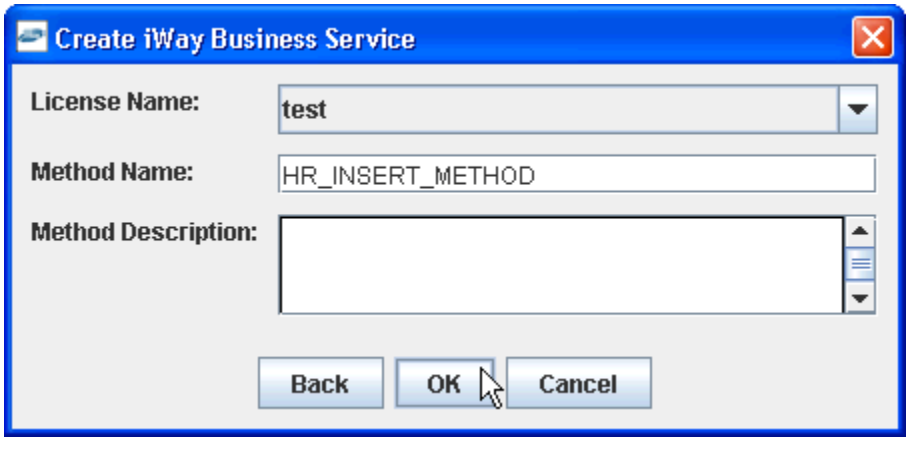
   a. From the Existing Service Names drop-down list, select whether you want to create a new service name or use an existing service name.

   b. In the Service Name field, type a descriptive name for the iWay Business Service.

   c. In the Service Description field, type a brief description of the service (optional).

5. Click *Next*.

   A second Create iWay Business Service dialog box opens and prompts you for additional information, as shown in the following image.



6. Perform the following steps:

   a. From the License Name drop-down list, select a license definition.

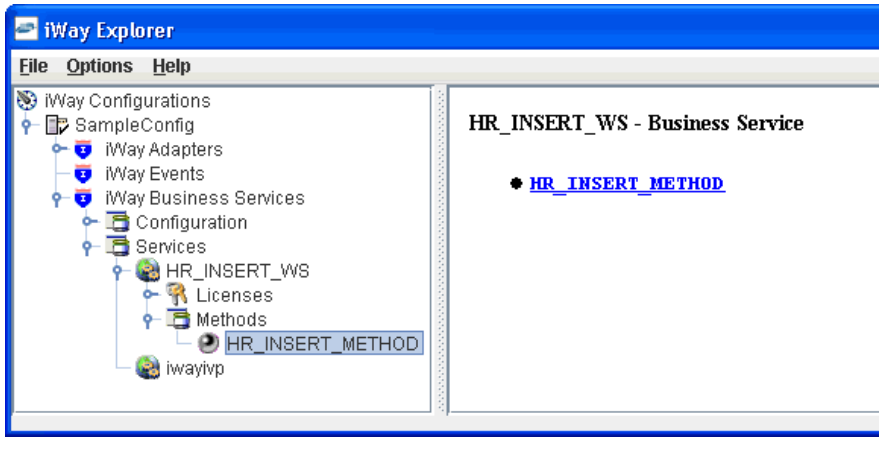
b. In the Method Name field, type a descriptive name for the method.

c. In the Method Description field, type a brief description of the method (optional).

7. Click *OK*.

The iWay Business Services node expands in the left pane. The new iWay Business Service appears under the Services node.

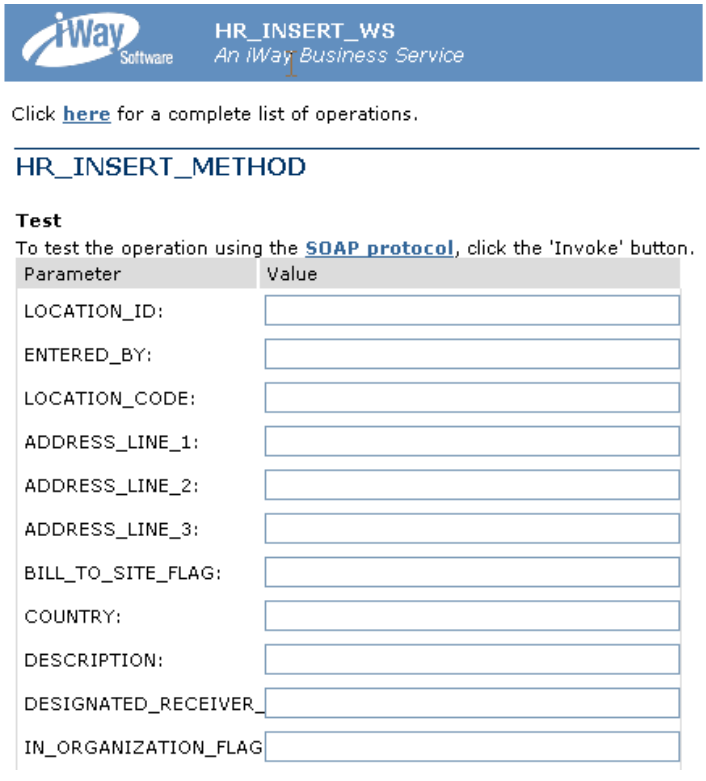
The following image shows an iWay Business Service called HR_INSERT_WS and, under that service, a method called HR_INSERT_METHOD.



The right pane displays the name of the expanded iWay Business Service and provides a hyperlink to the selected method, for example, HR_INSERT_METHOD.

8. Click the *HR_INSERT_METHOD* hyperlink in the right pane.

An iWay Business Service test pane opens in your web browser, as shown in the following image.



a. In the LOCATION_ID field, type the ID of the connection, for example, 390.

b. In the ENTERED_BY field, type the parameter, for example, 1.

c. In the DESCRIPTION field, type a descriptive name for the connection, for example, iWay Sample Test.

9. Click *Invoke*.

   The web service test result appears in the right pane.

   You have completed the configuration and testing of this service.

Chapter 9

Creating Prepared Statements and Batch Statements

This section describes how to create prepared statements and batch statements using iWay Explorer.

Although this section shows the Java Swing implementation of iWay Explorer, other implementations provide the same functionality by means of similar graphical user interfaces.

In this chapter:

Prepared and Batch Statement Overview

You can create an SQL statement even when using the adapter for non-relational databases. iWay Explorer provides the convenience to browse and view database tables while you create a statement. You can also edit existing SQL statements and parameters. The edit feature also allows you to view tables while you edit.

After you create the statement, you can generate schemas that define request and response documents. The metadata is stored in the iWay Repository, which can be implemented in an Oracle database.

You can generate the following types of statements:

- Regular SQL statements, as described in *Creating and Testing a Regular SQL Statement* on page 118.
- Parameterized SQL statements, as described in *Creating and Testing a Parameterized SQL Statement* on page 128.
- Batch Statements, as described in *Creating a Batch Statement and an Iterative Process* on page 145.

You can generate request and response schemas for:

- Regular (non-parameterized) SQL statements and parameterized SQL statements.
- Stored procedures for relational databases.

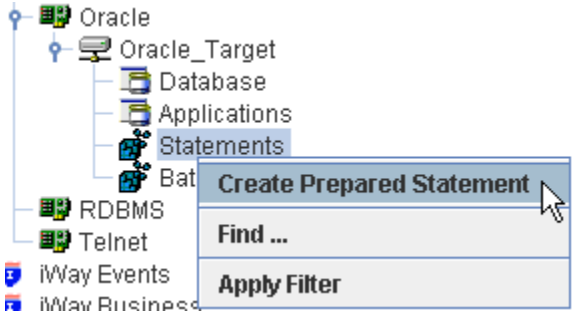
Creating and Testing a Regular SQL Statement

This section explains how to create and test a regular SQL statement. In addition, it describes how to edit an existing SQL statement.

Procedure: How to Create a Regular SQL Statement

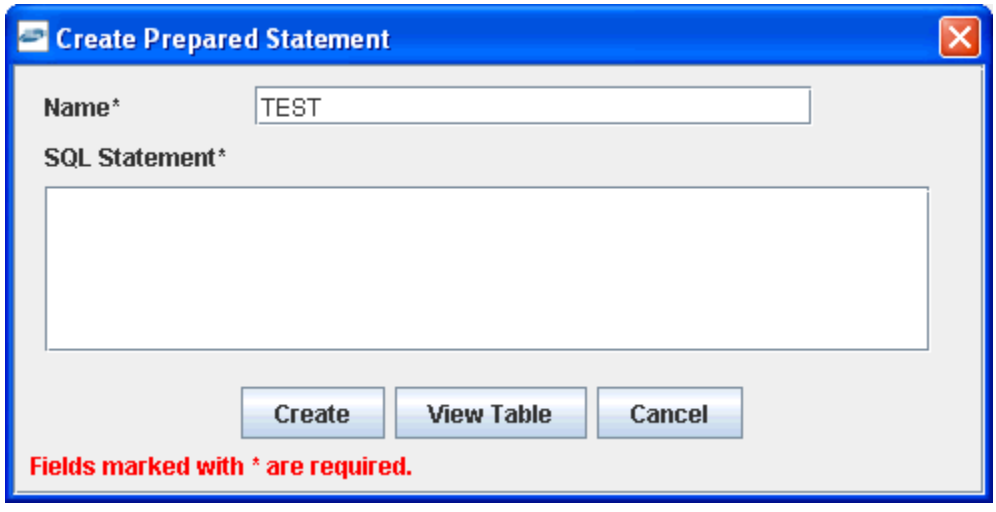
To create an SQL statement:

1. If you are not connected to a target, connect to one, as described in *Connecting to a Target* on page 62.
2. Click the *Statements* node.



3. Right-click the *Statements* node and select *Create Prepared Statement.*

The Create Prepared Statement dialog box opens, as shown in the following image.



4. In the Name field, type a name for the prepared statement.

   iWay Software recommends that you specify a name that describes the service. For example, a name of *CustomerIntField* could represent a request against the Customer Interface table returning a Field format response document.

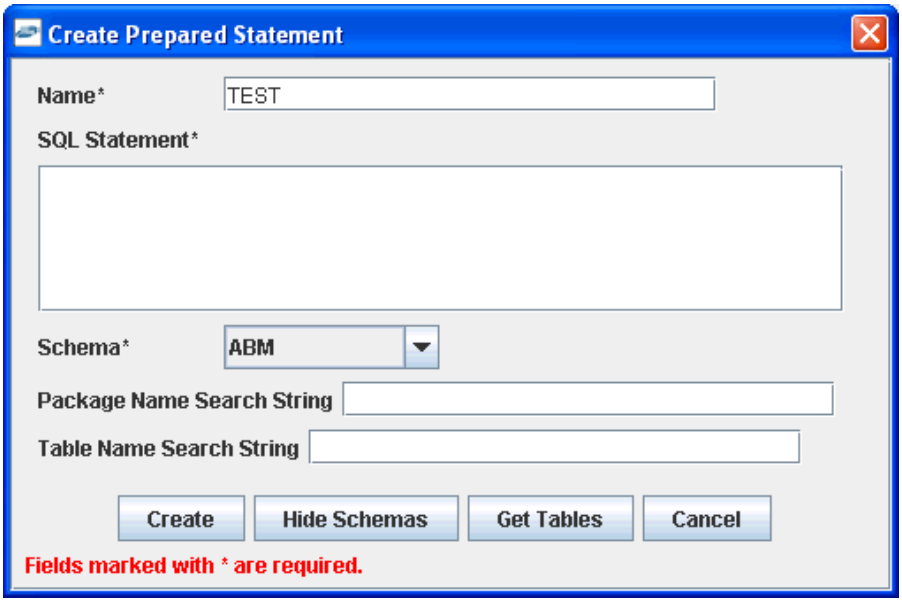
5. In the SQL Statement field, type the SQL statement for the adapter to use.

   Note: If you are not the owner of the table(s), the table name must be fully qualified.

   To view table metadata while you edit:

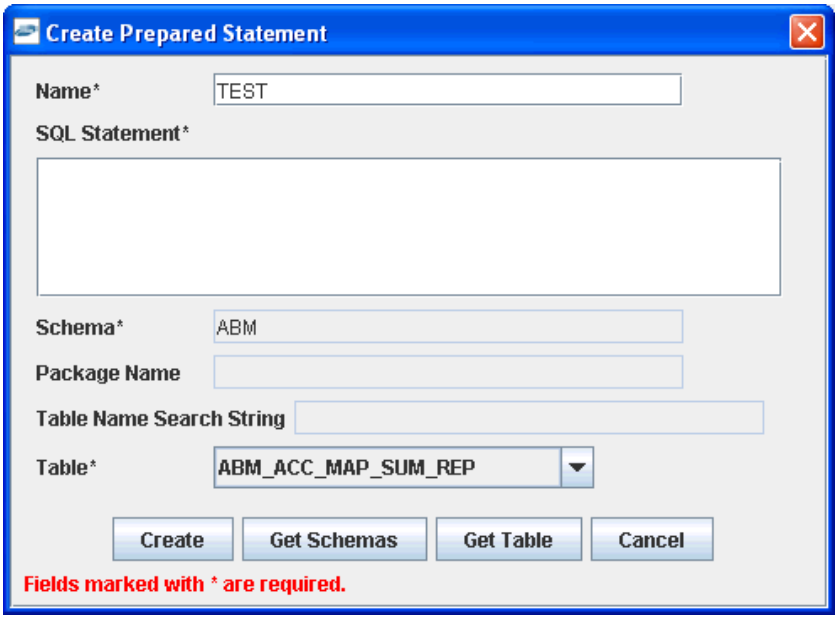
   a. Click *View Table*.

The Create Prepared Statement dialog box expands and the Schema drop-down list is now available.



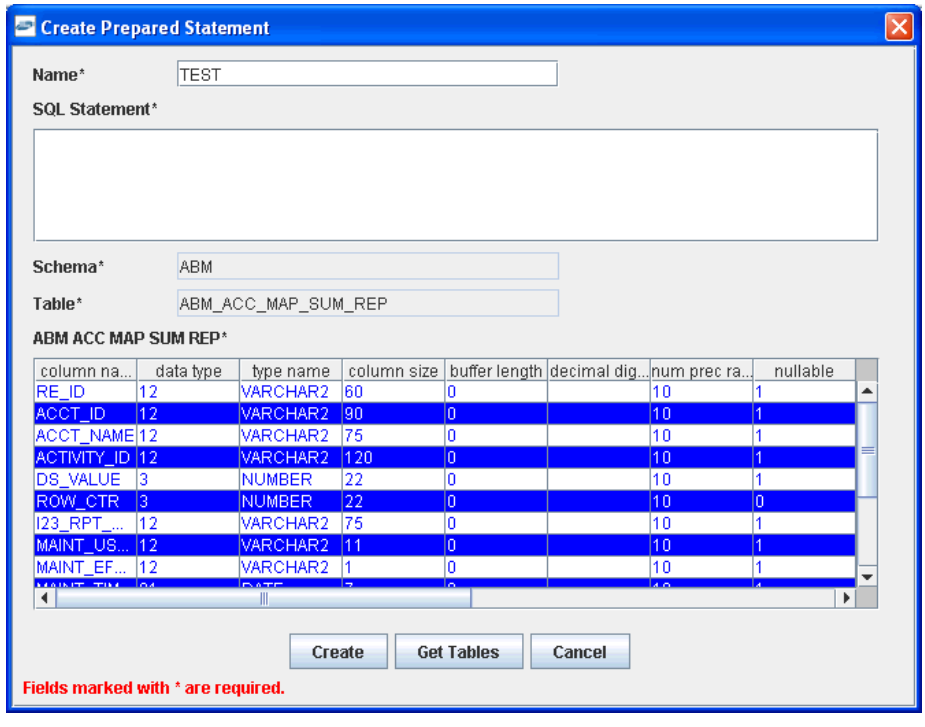
b. Select a schema from the Schema drop-down list and click *Get Tables*.

The Table drop-down list is now available under the Table Name Search String field.



c. Select a table you want to view from the Table drop-down list and click *Get Table*.

The table metadata appears at the bottom of the Create Prepared Statement dialog box. Use the horizontal and vertical scroll bars to view the entire table. An example of a table displayed in the Create Prepared Statement pane is shown in the following image.



You can browse additional tables if required by clicking the *Get Tables* button found at the bottom of this dialog box.

6. Enter an SQL statement in the SQL Statement field and click *Create*.

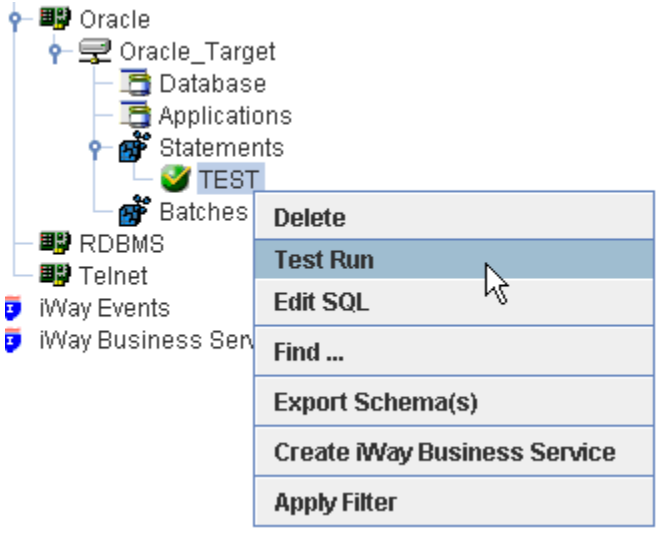
After the SQL statement node is built, you are ready to test the statement.

For information on testing a regular SQL statement, see *How to Test a Regular SQL Statement* on page 122.

Procedure: How to Test a Regular SQL Statement

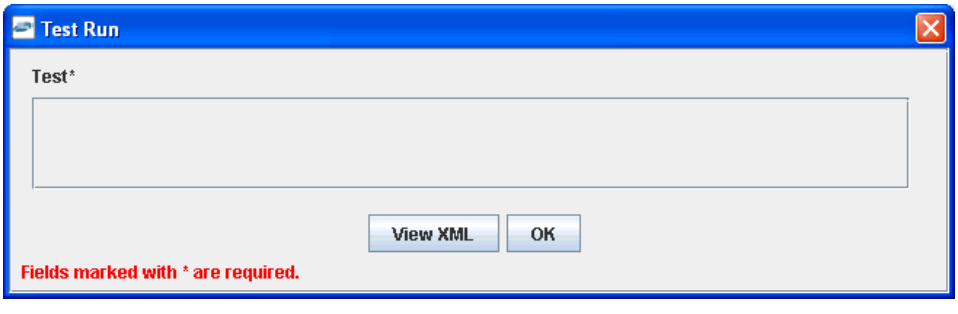
To test a regular SQL statement:

1. Select the SQL statement node you want to test.

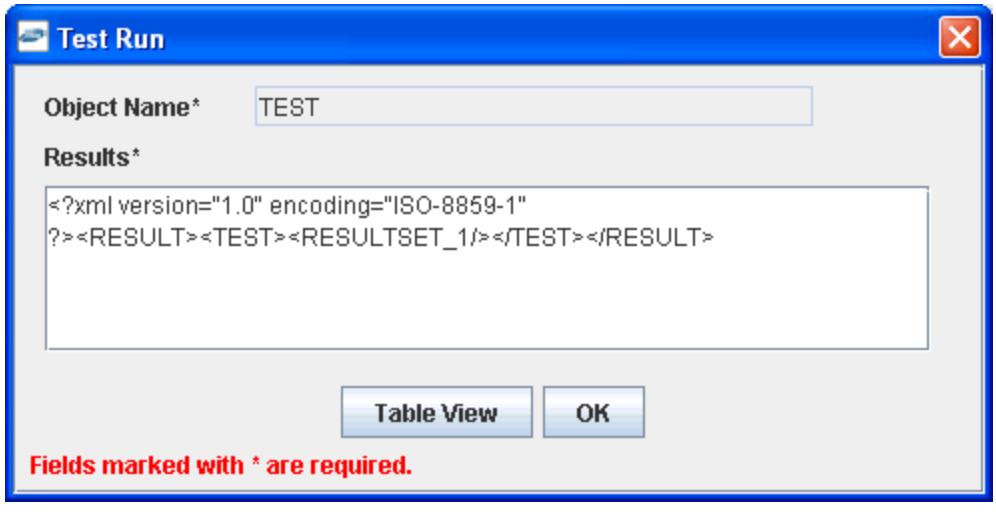


2. Right-click the SQL statement node and select *Test Run*.

   The Test Run dialog box opens and the test results appear in a table format, as shown in the following image.

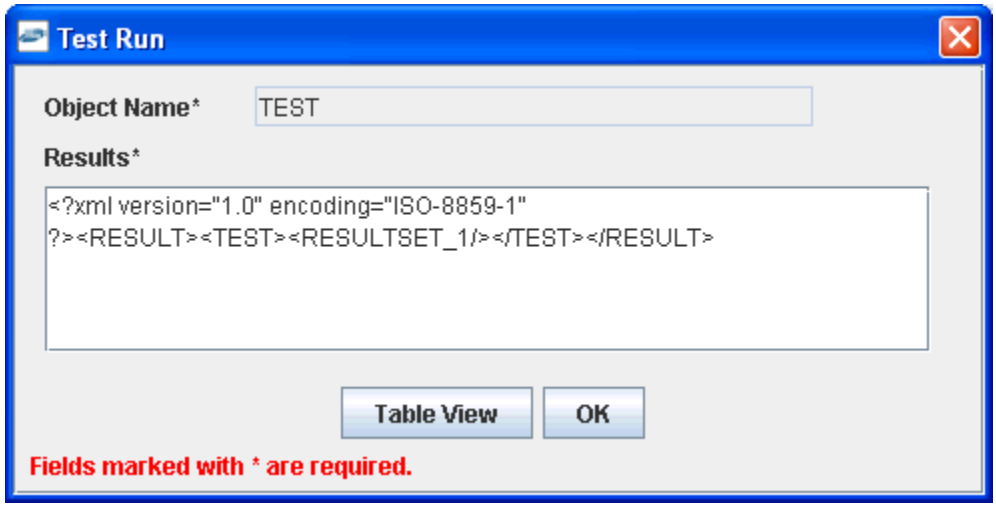


To see the results in XML, click *View XML*. The following image is an example of test results in XML format.



3. Click *Test*.

   To see the results in XML, click View XML. The following image is an example of test results in XML format.



   You can click *Table View* to return to the table format display.

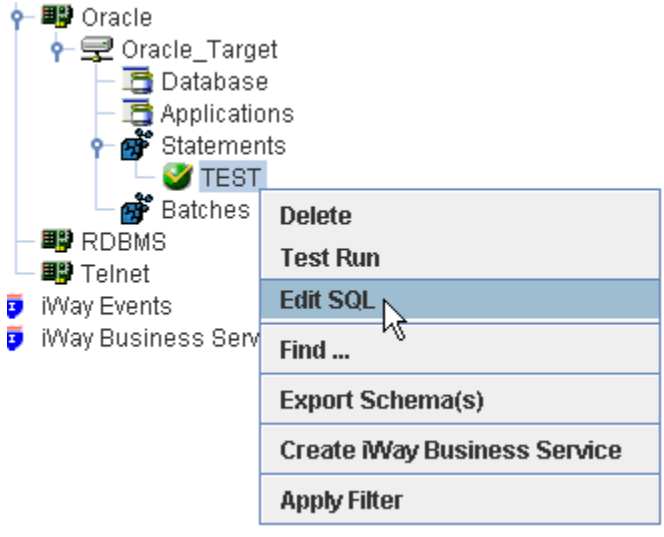
4. To exit the Test Run dialog box, click *OK.*

Procedure: How to Edit a Regular SQL Statement

Note: You can follow this procedure to edit the SQL for a parameterized SQL statement as well.

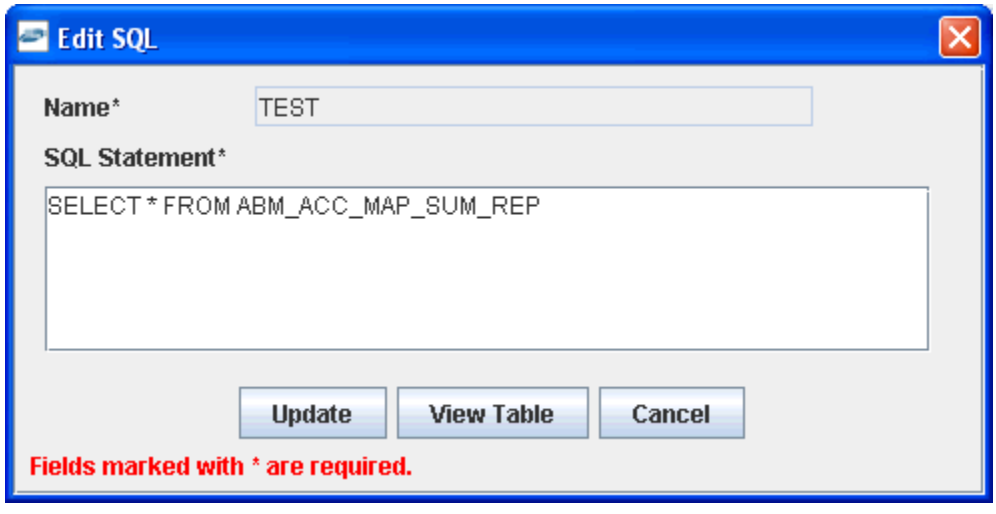
To edit an SQL statement:

1. Expand the *Statements* node in the left pane.
2. Select the SQL statement you want to edit.



3. Right-click the SQL statement node and select *Edit SQL*.

The Edit SQL pane opens on the right, as shown in the following image.



You can view table metadata by clicking *View Table*.

4. When your edits are complete, click *Update*.

About Table Functions

Table functions enhance the ability to update and query a table by creating commonly used prepared statements.

When you create web services from a table function, the adapter builds the SQL request and incorporates it into the web service. You can also export the schema and use it to generate XML instance request documents.

Existing APIs into which these functions can fit are the J2EE design pattern data accessobject (DAO), Java data objects (JDOs), and J2EE entity beans, all of which abstract andencapsulate access to a data source.

The functions include the following:

- SELECT
- CURSOR
- GET
- INSERT
- UPDATE

- DELETE
- UPSERT
- COUNT
- AVERAGE
- MAX
- MIN
- SUM

Procedure: How to Use the CURSOR Function

CURSOR is a query function that allows you to scroll through a result set without having an open cursor within the adapter.

1. Select the table node in which you are interested.
2. Select *CURSOR*.
3. Move the mouse pointer over *Operations* and select *Create iWay Business Service*.

   If you are using Swing iWay Explorer you can also Export the Schema by right-clicking the node in which you are interested and selecting *Export Schemas*. When using the JSP version of iWay Explorer, you can generate schemas by moving the mouse over Operations and selecting *Generate Schema*. You can use the schema to create instance XML request documents.

   The web service or schema that is created incorporates the SQL statement for the CURSOR function. This function requires five parameters. Only ROW_COUNT and ROW_REFERENCE require a value. The parameters are listed and defined in the table below.

Parameter	Description
ROW_COUNT (required)	The number of rows that you want the function to return. If you a supply a value of -1, all rows will be returned.

Parameter	Description
ORDERBY_COLUMN	The list of columns for the table. It is an enumeration and can have only values in its enumeration list. The column name passed to this parameter sets the “order by” clause in the dynamic statement generated by this function.
ROW_REFERENCE (required)	The row and all of its values from which the returned result set starts or ends. The function dynamically creates a select statement and determines the next set of rows to be sent based on the parameters sent.
ASCENDING	The boolean input parameter that determines if the “order by” is ascending or descending.
NEXT	The boolean input parameter that determines if the result returned is the next or previous set of rows.

Procedure: How to Use Test Run for the CURSOR Function

Using Test Run for the CURSOR function provides an opportunity to use the function. To use Test Run for the CURSOR function:

1. Ensure that the CURSOR function is selected.
2. Right-click the CURSOR function and select *Test Run*.

   The Test Run information appears.
3. Specify the information required:
   a. For Row count, enter the number of rows you want to be returned.
   b. From the Column drop-down box, select the column by which you want the result to be sorted.
   c. Select Ascending if you want to order rows in ascending order.
4. Click *Get Rows.*

   The result appears in the right pane.

5. Click *Previous* or *Next* to return the previous rows or next rows, respectively.

   The number or rows returned by clicking Previous or Next is the same number specified in the Row count parameter in the Test Run dialog box.

Creating and Testing a Parameterized SQL Statement

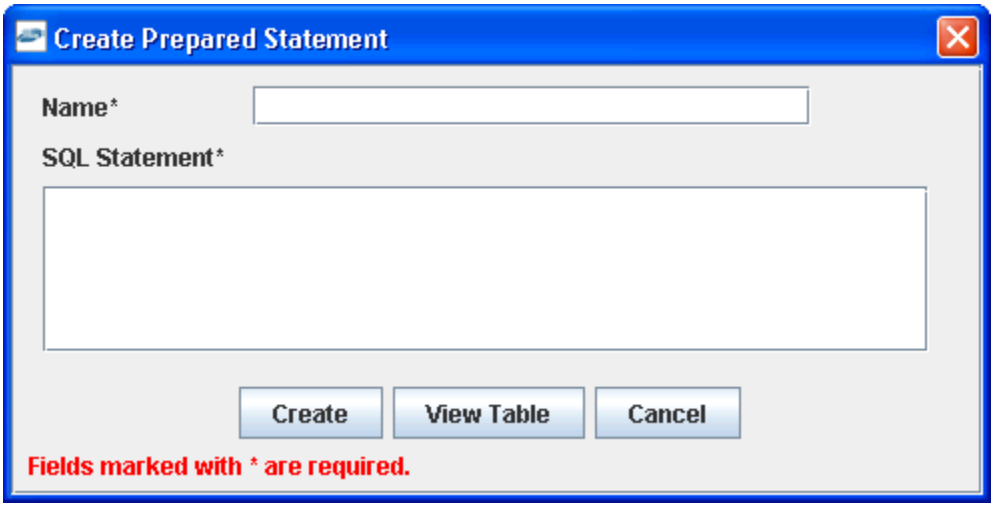
Parameterized SQL allows an SQL statement to be stored within the repository system with parameters imbedded within it. These parameters can be retrieved from XML documents at run time and executed against the SQL statements specified at design time. iWay Explorer creates and maps parameters for the parameterized SQL at design time.

Procedure: How to Create a Parameterized SQL Statement

To create a parameterized SQL statement:

1. Connect to a defined target.
2. Click the *Statements* node.
3. Right-click the *Statements* node and select *Create Prepared Statement*.

   The Create Prepared Statement opens, as shown in the following image.



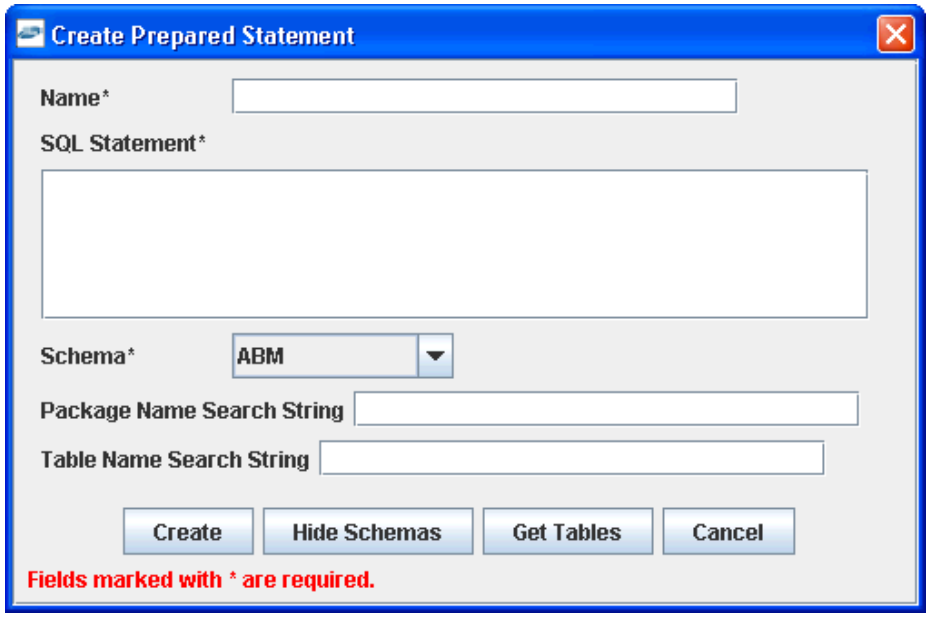
4. In the Name field, type a name for the statement.
5. In the Enter SQL Statement field, type the parameterized SQL statement.

   Note: If you are not the owner of the table or tables, the table name must be fully qualified.

To view table metadata while you edit:

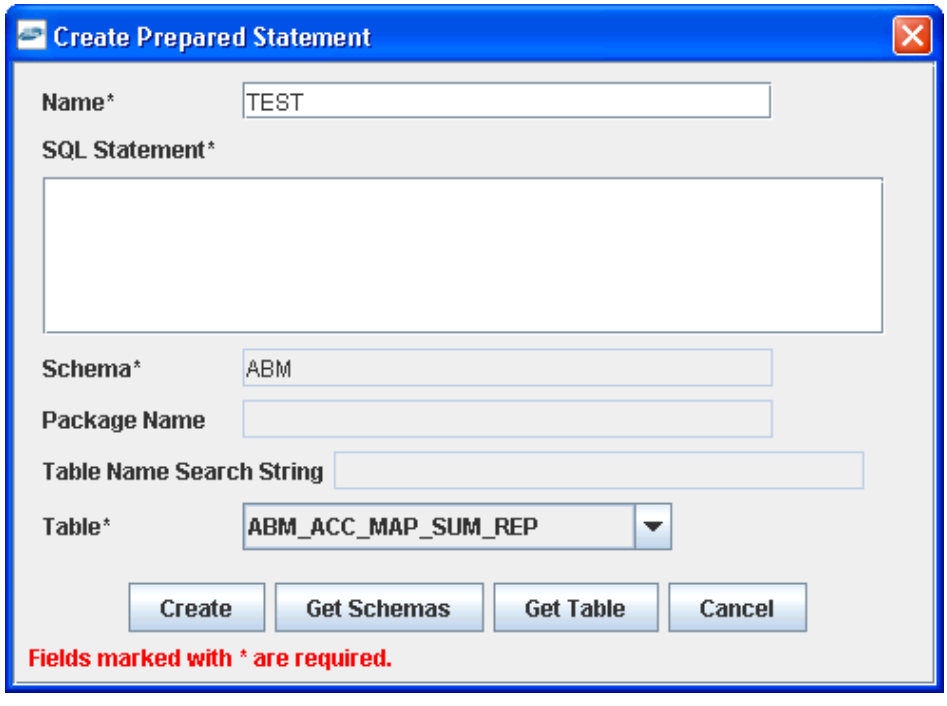
a. Click *View Table*.

The *Schema* drop-down list is now available at the bottom of the pane.



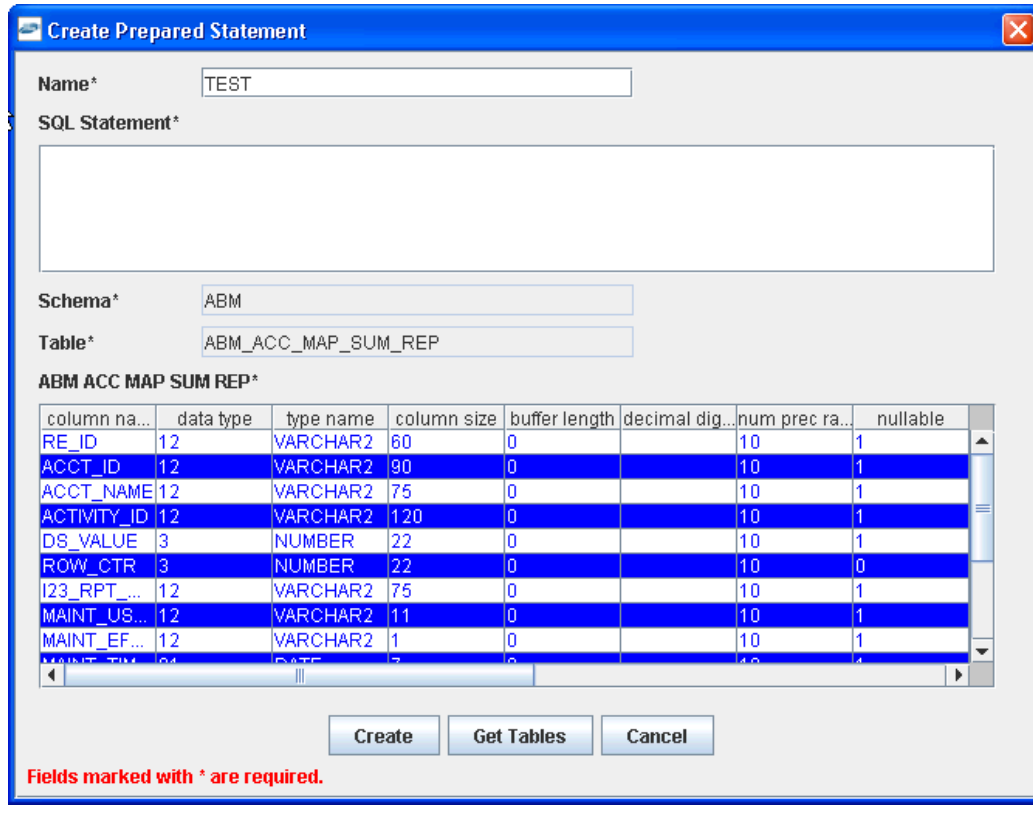
b. Select a schema from the drop-down list and click *Get Tables*.

The *Table* drop-down list is now available under the *Schema* field.



c. Select the table you want to view from the drop-down list and click *Get Table*.

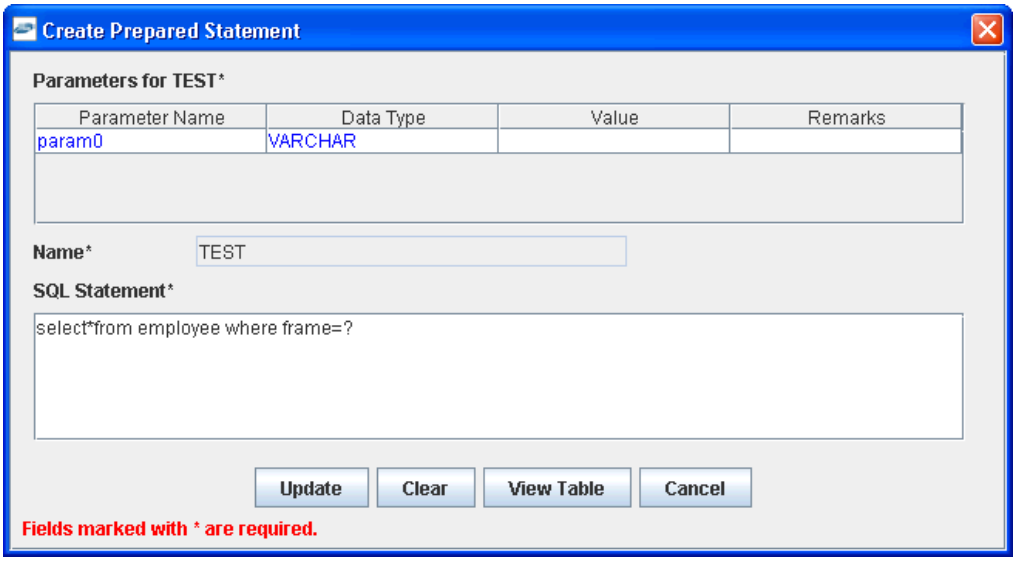
The table metadata appears. Use the horizontal and vertical scroll bars to view the entire table. An example of a table displayed in the Create Prepared Statement pane is shown in the following image.



You can browse additional tables if needed using Get Tables found at the bottom of this pane.

6. When the parameterized statement is complete, click *Create*.

The Parameter Name, Data Type, and Value selection information appears at the bottom of the pane, as shown in the following image.



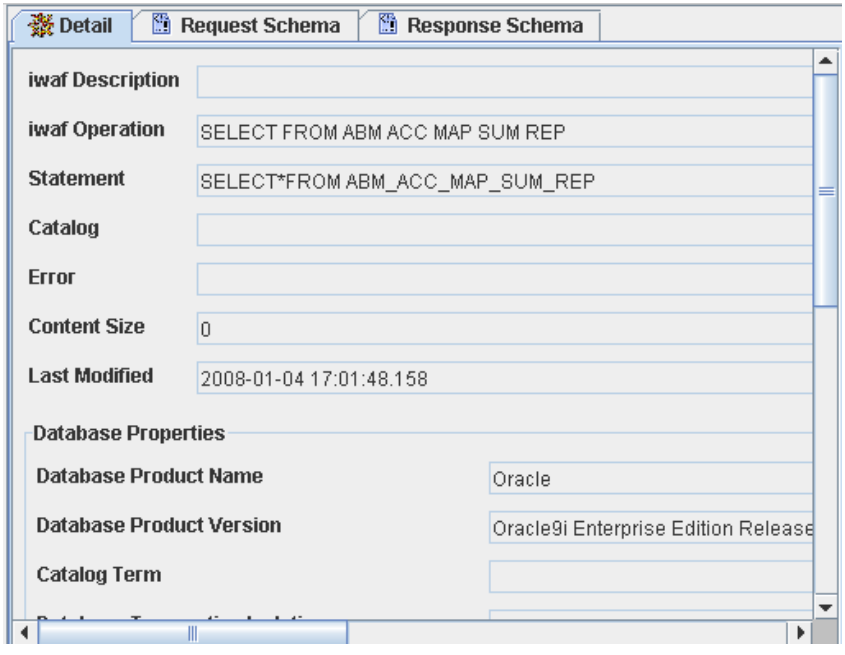
a. In the Parameter Name column, type a name for each parameter.

b. In the Data Type column, select a data type for each parameter from the drop-down list.

c. In the Value field, type a value for the parameter.

7. Click *Update*.

The information for the newly created statement appears in the right pane as shown in the following image.



Use the horizontal and vertical scroll bars to view the properties and parameters.

If your statement is not accurate, an error message appears at the bottom of the pane.

The date on which the statement is created is saved along with other data about the statement. In addition, the date for each parameter is saved. The date information can help debug problems. For example, if a batch statement that references a prepared statement has a modified date earlier than the date listed for the prepared statement, the batch might behave differently than expected. Also, when the design-time and runtime repositories are the same, a deployed service with a date earlier than the modified date shown for the service in design-time might mean that the service behaves differently than intended. In this case, the service should be redeployed.

For information on testing a parameterized SQL statement, see *How to Test a Parameterized SQL Statement* on page 133.

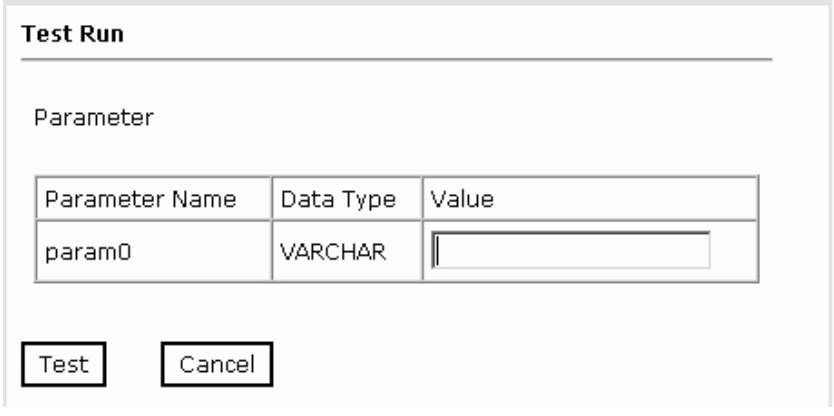
Procedure: How to Test a Parameterized SQL Statement

To test a parameterized SQL statement:

1. In the left pane, select the parameterized SQL statement node you want to test.

2. Right-click the statement and select *Test Run*.

   The Test Run pane opens on the right for the SQL statement. This pane contains the parameter name, data type, and an input box where you can type the parameter value, as shown in the following image.



3. For each parameter, type a value in the Value field.

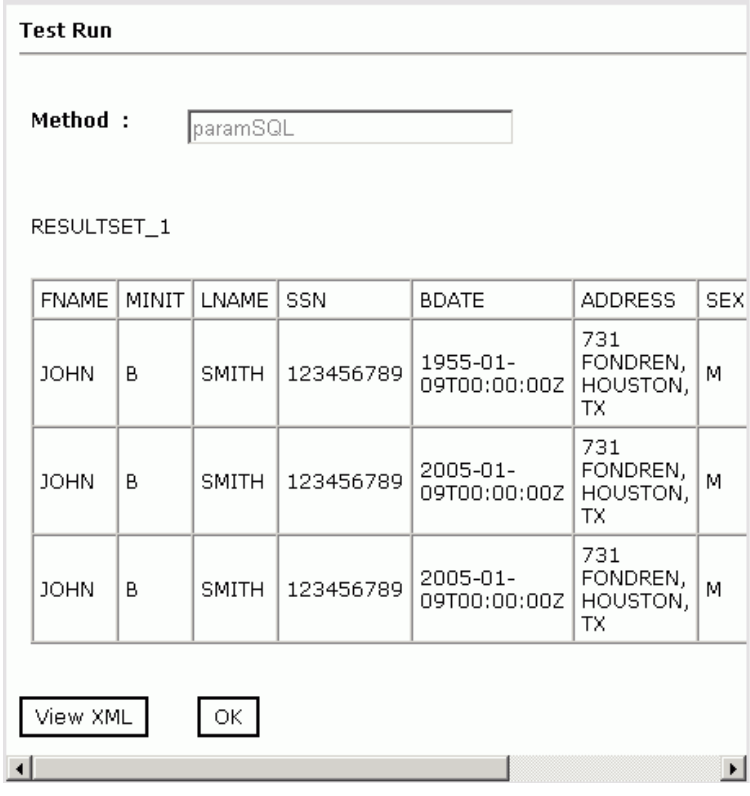
   For example, provide a sample character value, for example, BELLA, for the following SQL statement:

   ```
   select * from employee where lname=?
   ```

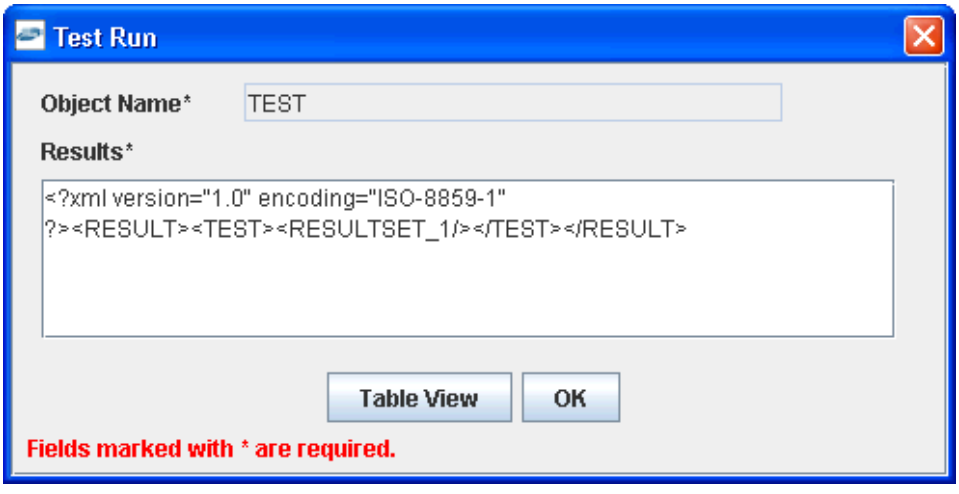
   In this example, the values correspond to values of fields found in a table. Parameterized statements may include parameters that are input for SQL functions, for example, the Oracle SQL function TO_DATE(StringParm). In this case, the data type selected is the expected data type of the SQL function. This is why you provide the SQL type when you create the prepared parameterized SQL statement.

4. Click *Test*.

The results appear in the Test Run results window in a table format. An example of test results is shown in the following image.



To see the results in XML, click *View XML*. The following image is an example of test results in XML format.



Click *Table View* to return to the table format display.

5. To exit the results, click *OK*.

Procedure: How to Edit a Parameterized Statement

You can edit the parameter name, data type, and value through the Edit Parameters pane.

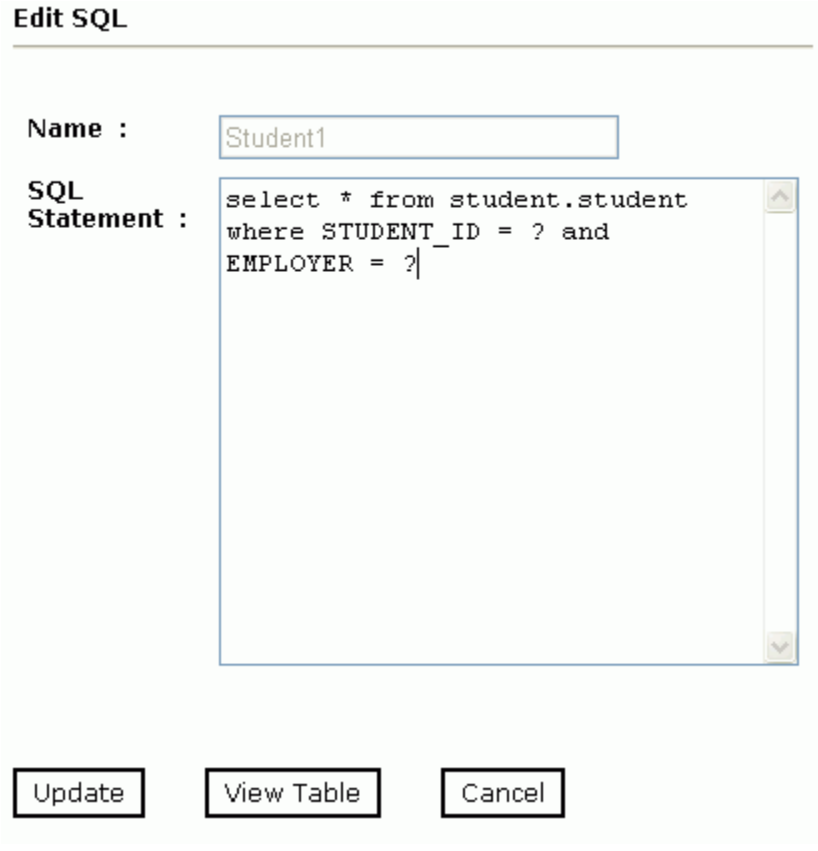
To edit a Parameterized SQL statement:

1. Expand the Statements node in the left pane.
2. Select the Parameterized SQL statement you want to edit.
3. Right-click the statement and select *Edit Parameters*.
4. Type the changes to parameter name and value, and select the data type as necessary.

   To view table metadata while you edit, click View Table. For more details see *How to Create a Parameterized SQL Statement* on page 128
5. When your edits are complete, click *Update*.

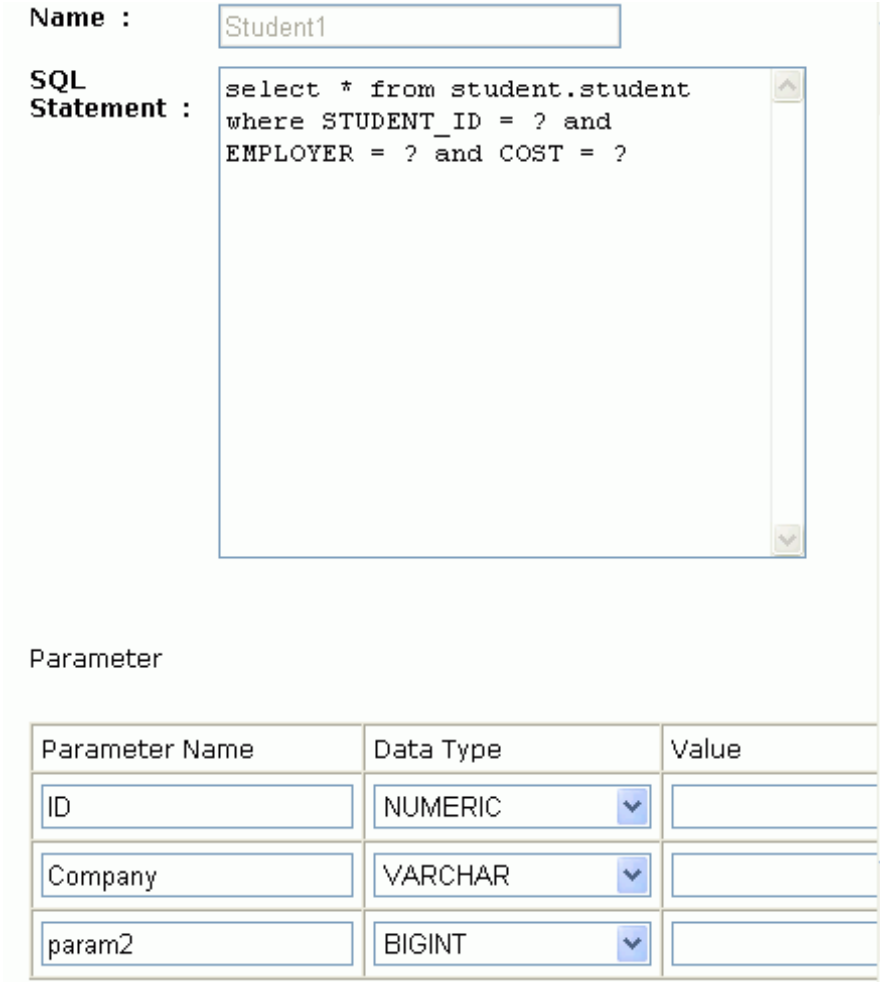
   If your statement is not accurate, an error message appears at the bottom of the pane.
6. If you need to add or delete a parameter, right-click the parameterized statement, and select *Edit SQL*.

For example, the following image shows a parameterized statement before it is edited:



7. Edit the SQL to add the new parameter, for example, COST, and click *Update*.

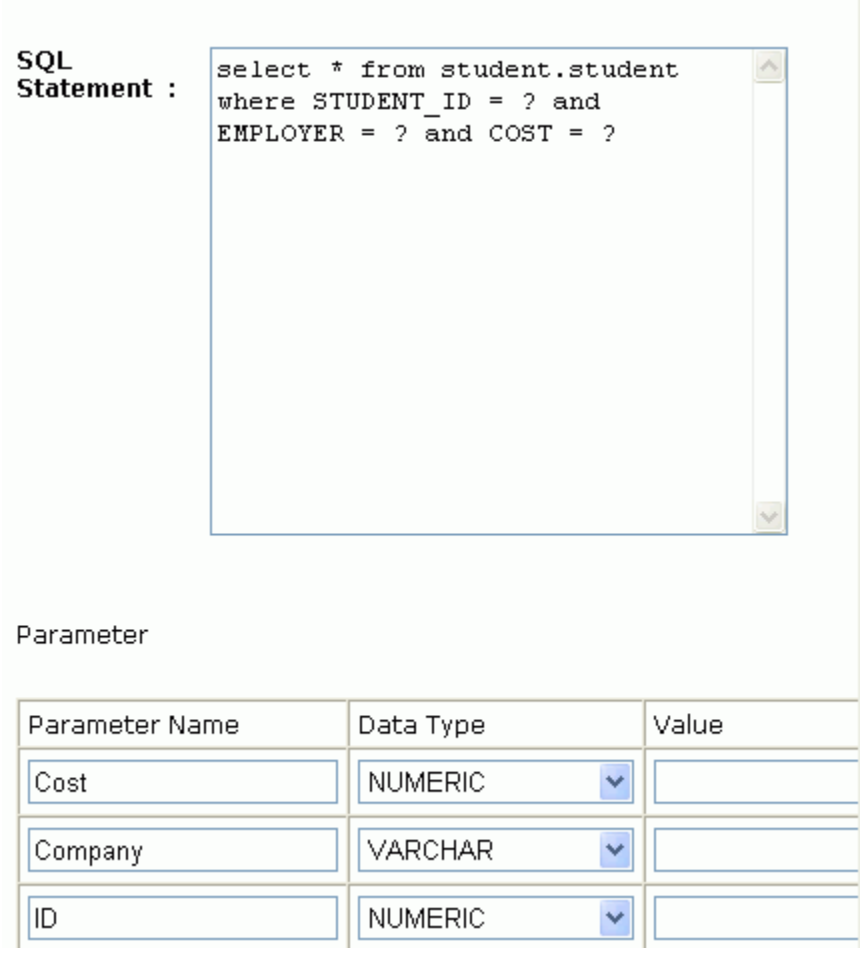
When a new parameter is added to the SQL statement, iWay Explorer adds the new, unnamed parameter automatically to the parameter list. The following image shows the edited SQL statement and the parameters listed below the SQL Statement box:



You can edit the parameter name and data type.

8. Edit the parameter as needed and click *Finish* or click *Update* if you are not done editing the statement.

The following image shows the edited parameter in the parameter table.



You can change the position of the parameters in the parameter table to match the order of parameters in the SQL statement by specifying the position in the Position/Delete column. The following image shows the reordering of the parameter rows in the Parameter table based on the position selected. As shown in the image, the third row will become the top row, followed by the two other rows.

9. Click *Change* to view the changes in the parameter table without committing them, or click *Finish* to commit the changes.
10. To delete a parameter, edit the SQL to remove the parameter and click *Change*.

    The delete setting appears by default in the Position/Delete column of one of the parameter rows. If it is not the parameter you want to delete, you can change the setting for that row.
11. Select *delete* from the Position/Delete column for the parameter you are deleting.

    The following image shows delete selected in the second row, which contains the Company parameter:

You can also change the position of a parameter relative to other parameters in the list by selecting a position number in the parameter row whose position you want to change. This is useful if you delete the middle parameter from a SQL statement and want to reposition the other parameters in the parameter table to match the sequence of parameters in the SQL statement. The following image shows delete selected in the middle row and a position of 1 selected in the third row of the parameter table. This will move the position of the this parameter from the third row to the second row:

12. Click *Change* to perform the modification without committing the changes, or click *Finish* to complete the edits and commit the changes.
13. If you want to clear the edits and return the parameter list to its original form, click *Undo Edit*.
14. If you want to remove all parameters from the list and start fresh, click *Clear All*.
15. If you are not satisfied with the edits, click *Undo Edit* to clear the edits you are making and return the parameter list to its original form or click *Clear All* to return the list to a list of unnamed parameters.
16. Click *Finish* when you are done with all the edits and want to commit the changes.

Using Date and Time Formatting

Because dates can be formatted in many different ways, some applications might have to take extra time transforming a date and time string to the correct XML format. You can specify the date and time format when you design a service so that it does not have to be done in the flow of the message. You can configure and test the date formatter the adapter uses at runtime to parse the request value.

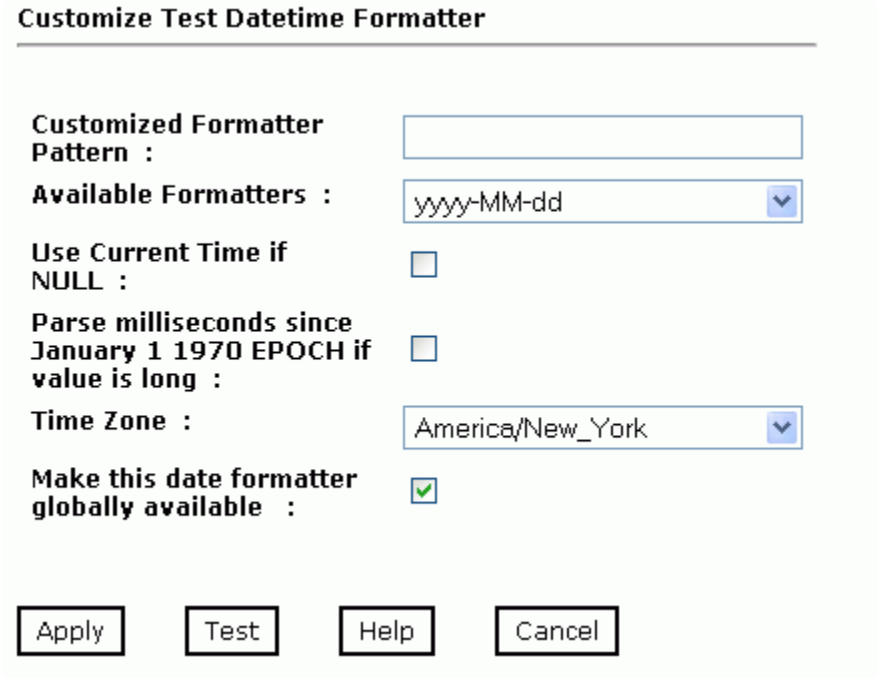
Note: The date format you specify structures the date format for the request document. The date format returned in the response will conform to the date format defined in the database.

Procedure: How to Use Date and Time Formatting Options

To specify the date and time format in a request:

1. Select the date parameter in the prepared SQL statement for which you want to format the date and time.
2. Right-click the parameter and select *Customize Test Datetime Formatter*.

   The Customize Datetime Formatter pane appears on the right, as shown in the following image:

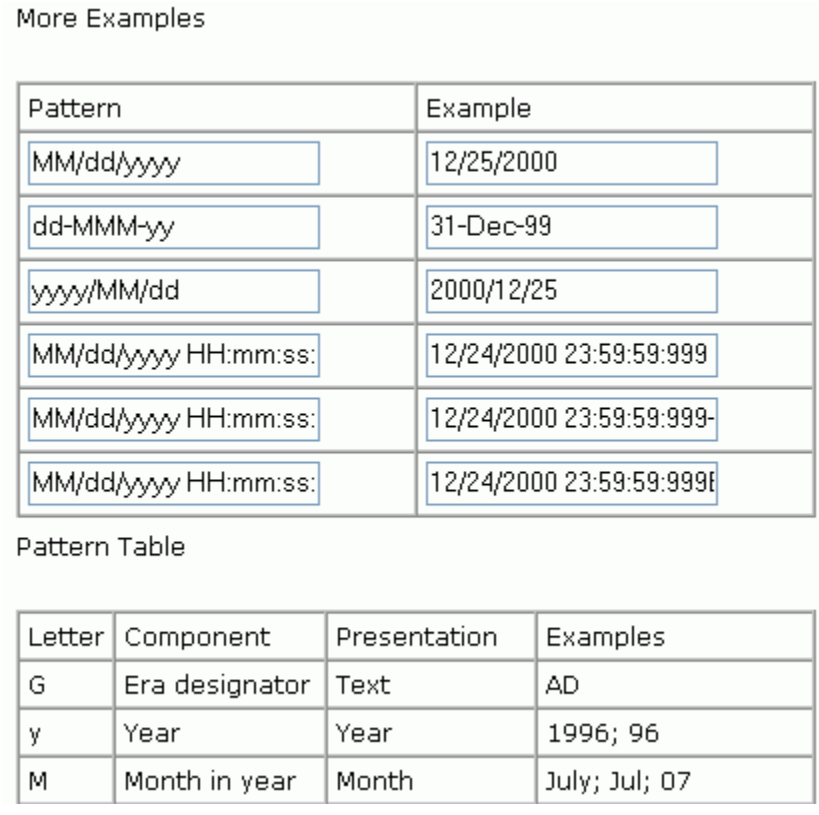


3. Provide the information as follows:

   a. In the Customized Formatter Pattern box, provide the date and time pattern, or select a Formatter from the Available Formatters drop-down box.

   b. Click *Help* if you want to see two tables that assist you in configuring the custom formatter, the Pattern Table and an examples table.

The Examples and Pattern tables appear. The following image shows the Examples Table and the Pattern Table. The Examples Table lists date examples. The Pattern Table specifies the letter, component of the date, the date presentation, and examples.



More Examples

Pattern	Example
MM/dd/yyyy	12/25/2000
dd-MMM-yy	31-Dec-99
yyyy/MM/dd	2000/12/25
MM/dd/yyyy HH:mm:ss:	12/24/2000 23:59:59:999
MM/dd/yyyy HH:mm:ss:	12/24/2000 23:59:59:999-
MM/dd/yyyy HH:mm:ss:	12/24/2000 23:59:59:999[

Pattern Table

Letter	Component	Presentation	Examples
G	Era designator	Text	AD
y	Year	Year	1996; 96
M	Month in year	Month	July; Jul; 07

c. Select *Use Current Time if NULL* to use the current time as the default value if the request value is null.

d. Select *Parse milliseconds since January 1, 1970 (EPOCH)* if you want to have the adapter check if the value returned is a long value representing the number of milliseconds since January 1, 1970 (EPOCH).

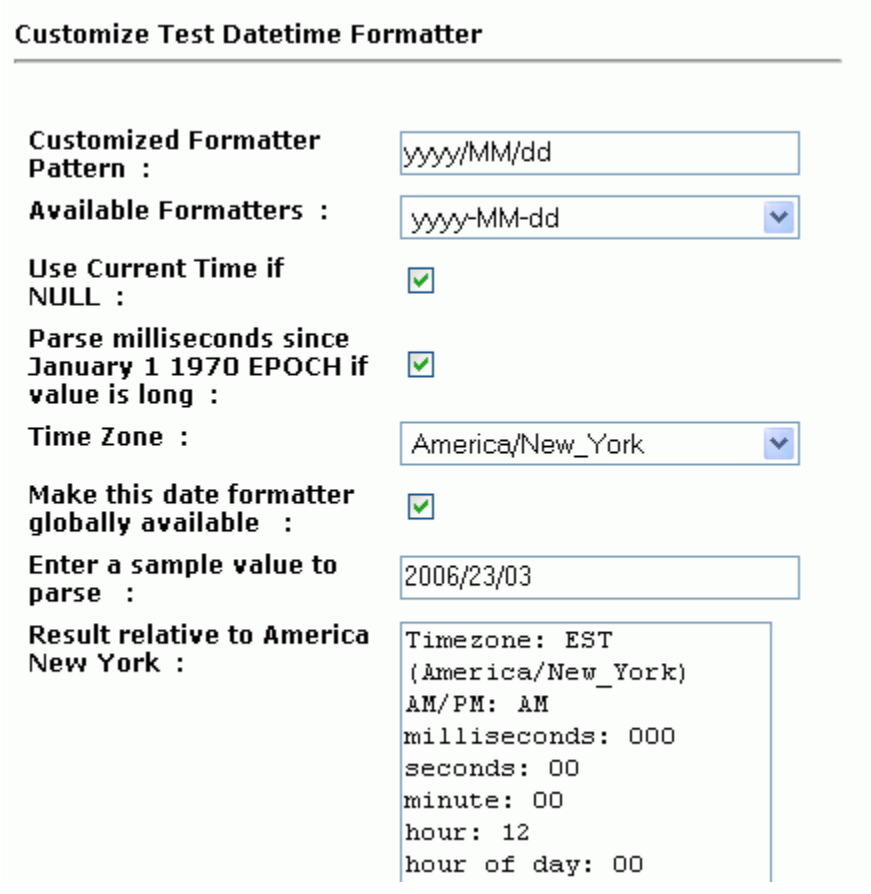
e. Select the time zone from the Time Zone drop-down list.

f. Select *Make this date formatter globally available* if you want this formatter available for use by other services.

g. To test a sample date value, click Test and enter a sample value to parse in the *Enter a sample value to parse* text entry box.

h. Click *Test* to test the sample date value.

The following image shows sample test results.



4. Click *Apply* to commit the changes.

Executing an SQL Statement or Stored Procedure Multiple Times

You can execute a prepared SQL statement or stored procedure multiple times with different input parameters each time. The major benefits of this feature are as follows:

- **Connection Resource Utilization.** One connection or thread is established to the back-end database. This minimizes resource consumption. Note that the number of cursors for select statements created is based on the number of parameter sets in the submitted request. If there are three sets of parameters for a select statement, there will be three cursors created and closed sequentially.

- **Logical Unit of Work (LUW).** A logical unit of work (LUW) is established that will roll back all of the updates or inserts that were issued by the SQL statement of prior executions. A transaction starts at the first set of parameters. The sets of parameters are executed in order from top to bottom of the message sequentially. For example, if an insert statement is submitted along with three sets of parameters, and the third set of parameters fails to insert, the previous two inserts will be rolled back.

- **Integration of Data From an Outside Source.** For integration scenarios where external data is used to "feed" SQL or stored procedures, this feature allows external data to be mapped to one XML execution block for the adapter to execute.

Note: Multiple processes in one message is not supported. Only one SQL statement or stored procedure can be executed in a single XML execution request block.

Creating a Batch Statement and an Iterative Process

Batch statements enable you to execute multiple SQL and/or parameterized SQL statements within one transaction.

Iterative processes are batch type processes that reference other processes, including stored procedures. In an iterative batch statement, each statement and procedure can be called multiple times. This differs from a regular batch statement, in which a statement or procedure can be called in a single batch; instead, an iterative batch statement allows you to iterate the batch itself.

If you add a statement to a batch without selecting the Iterate option, you can call the statement only once. If you iterate the statement only, which is done by default, you can iterate only a single statement. By choosing the Iterate option when you add a statement to a batch, you will be able to call several statements multiple times from a single batch.

All iterative processes reside within the Batches node. All prepared statements created in the Statements node are automatically imported into the Iterate container. However, you can import processes from Tables or stored procedures as well.

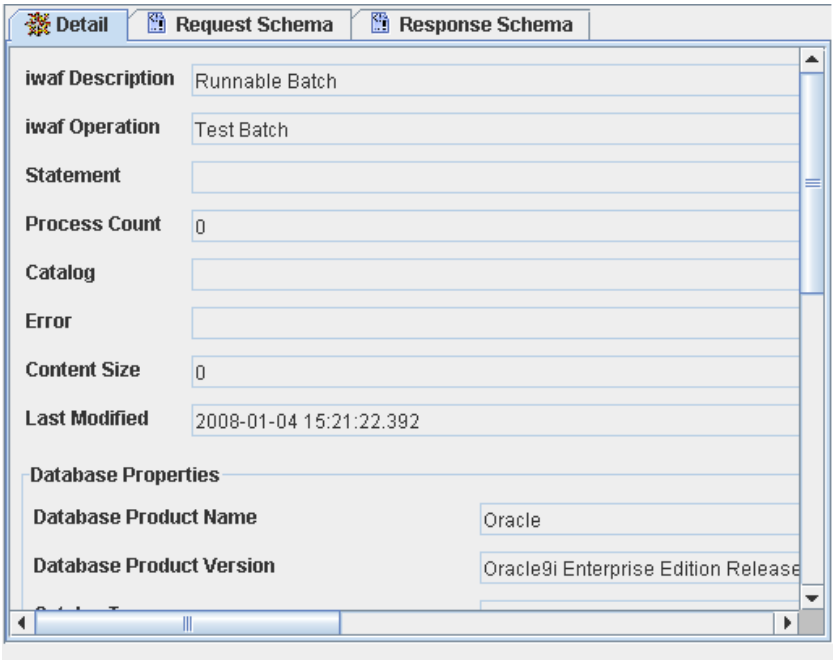
Procedure: How to Create a Batch Statement

To create a batch statement:

1. Connect to a defined target.
2. In the left pane, select the Batches node.
3. Right-click the Batches node and select *Create A Batch*.

   The Create A Batch pane opens.
4. Type a name for the new Batch and click *Create*.

The batch properties information appears in the right pane as shown in the following image.



5. Right-click the batch and select *Edit Batch*.

   The Node Type drop-down selection appears.

6. From the drop-down list, select the statement, table, or procedure and click *Next*.
7. Select *Iterate* if you want to make the procedure, statement or table iterative.
8. To add more statements or procedures, select and right-click the batch node in the left pane and select the appropriate option.

Procedure: How to Create An Iterative Process

To import tables or processes into an iterative process:

1. Connect to a defined target.
2. Expand the Batches node.
3. Select *Iterate*.

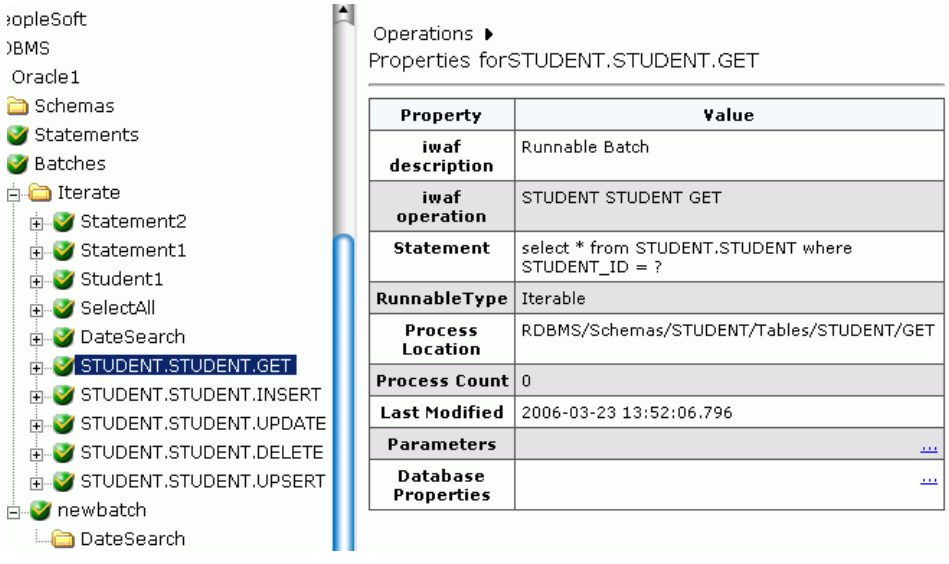
   By default, the Iterate node contains all the statements created in the Statements node.

4. Right-click the Iterate node and select *Import Process*.

The Import Process pane appears.

5. Select either Tables or Procedures to import and click *Next*.
6. Select Schema and click *Next*.
7. Select the tables or procedures you want to import and click *Next*.

   If you import tables, the GET, INSERT, UPDATE, DELETE, and UPSERT (but not CURSOR) functions are imported for each table you select. The following image shows the addition of the table functions for the STUDENT table added to the Iterate node:



   The stored procedures you Import will appear the same way.

Example: Creating an Iterative Process

The following is an example of the input XML for a batch statement named batchtest. Two parameterized SQL statements were added to this statement, one named ParamSTMT1, with an INSERT accessing TableA, and another named ParamSTMT2, with an INSERT accessing TableB. When the SQL statements were added to the batch statement, the Iterate check box was selected.

In this input XML document, two different statements are being called, with two different sets of values for each statement.

Note: Added statements or procedures to a batch must be included in the input document.

```
<?xml version="1.0" encoding="UTF-8"?>
<RDBMS xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\schemas\ora013_request.xsd">
    <BATCH location="RDBMS/Batches/batchtest">
     <ParamSTMT1 location="RDBMS/Batches/Iterate/ParamSTMT1">
       <PARAMS>
         <lname>Chunnulal</lname>
         <age>30</age>
       </PARAMS>
       <PARAMS>
         <lname>Doe</lname>
         <age>31</age>
       </PARAMS>
     </ParamSTMT1>
     <ParamSTMT2 location="RDBMS/Batches/Iterate/ParamSTMT2">
       <PARAMS>
         <fname>Neena</fname>
         <empid>4</empid>
       </PARAMS>
       <PARAMS>
         <fname>John</fname>
         <empid>5</empid>
       </PARAMS>
     </ParamSTMT2>
    </BATCH>
</RDBMS>
```

The following is an example of the input XML for a regular ITERATE process.

```
<RDBMS>
    <ITERATE location="RDBMS/Batches/Iterate/ParamSQL1">
          <PARAMS>
               <param0>a</param0>
               <param1>b</param1>
          </PARAMS>
          <PARAMS>
               <param0>c</param0>
               <param1>d</param1>
          </PARAMS>
    </ITERATE>
<RDBMS>
```

Procedure: How to Add Processes to a Batch Process

You can add a new process to a batch statement after you create it. The adapter also allows you to add a process to a batch even if the batch already contains it, allowing you to make limitless calls to the same process.

1. Connect to a defined target.
2. Expand the Batches node.
3. Select the batch statement you want to edit.
4. Right-click the statement and select *Edit Batch*.

   The Edit Batch pane appears on the right.
5. Select the type of process you want to add from the Node Type drop-down box, and click *Next*.
   a. If you select **Tables**, select the schema and then the particular table, and then the table function, clicking Next after each selection. After selecting the table function, select *Iterate* to iterate the table *function*, and then click *Add*.

      The iterate setting determines whether the table function you choose will be iterative. If so, the adapter searches for the process node in the Iterate node. If not, the adapter will search for it within the Schemas node.

   b. If you select **Statements**, select the statement you want to add from the Prepared Statement drop-down box, select Iterate if you want the prepared statement to be iterative, and click *Add*.

      You have the option of viewing the details of the statement you are adding by clicking *View Details*.

      The iterate setting determines whether the prepared statement you choose will be iterative. If so, the adapter searches for the process node in the Iterate node. If not, the adapter will search for it within the Statements node.

   c. If you select **Procedures**, select the schema from the Schema drop-down list, and click *Get Stored Procedures*. Select the stored procedure you want from the Stored Procedure drop-down box, select *Iterate* if you want the Procedure to be iterative, and click *Add Stored Procedure*.

      You have the option of viewing the details of the procedure you are adding by clicking *Show Details*.

      The iterate setting determines whether the Procedure you choose will be iterative. If so, the adapter searches for the process node in the Iterate node. If not, the adapter will search for it within the Procedures node.

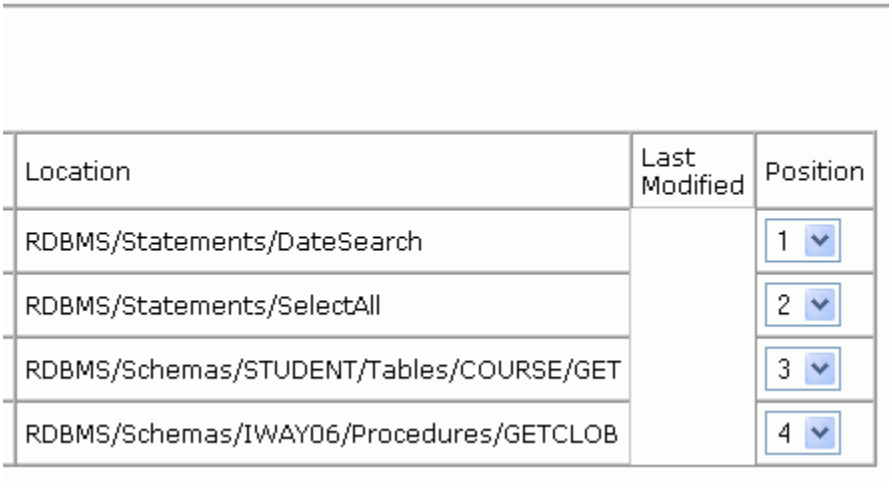
Procedure: How to Reorder Processes in a Batch Process

You can reorder the processes within a batch statement:

1. Connect to a defined target.
2. Expand the Batches node.
3. Select the batch statement you want to edit.
4. Right-click the statement and select *Edit Batch*.

   The Edit Batch pane appears.
5. In the right pane, click *Reorder Processes*.

   A table listing the processes appears in the right pane. Each row includes a position column.



Location	Last Modified	Position
RDBMS/Statements/DateSearch		1
RDBMS/Statements/SelectAll		2
RDBMS/Schemas/STUDENT/Tables/COURSE/GET		3
RDBMS/Schemas/IWAY06/Procedures/GETCLOB		4

6. Select the position for each row as appropriate and click *Reorder* to review the new order.
7. Click *Commit* to save the changes.

Procedure: How to Test a Batch Process

To test a batch process:

1. Connect to a defined target.
2. Expand the Batches node.
3. Select the batch statement you want to edit.
4. Right-click the statement and select *Test Run*.

The Test Run pane appears on the right. You can edit parameter names and data types for the test.

5. Click *Test*.

   The results appear in the right pane.

Chapter 10

Listening for Oracle E-Business Suite Events

This section describes how to listen for Oracle E-Business Suite events. Several listening techniques are available, enabling you to choose the technique that best suits your requirements.

Although this section describes the Java Swing implementation of iWay Explorer, other implementations provide the same functionality by means of similar graphical user interfaces.

In this chapter:

Understanding iWay Event Functionality

Events are generated as a result of Oracle E-Business Suite activity. You can use events to trigger an action in your application. For example, you can detect WIP discrete job creation and notify another enterprise integration system of the event.

After you create a connection to your application system, you can add events using iWay Explorer. To create an iWay event, you must create a port and a channel.

- **Port**

  A port associates a particular business object exposed by an adapter with a particular disposition. A disposition defines the protocol and location of the event data. The port defines the end point of the event consumption.

  For example, you can use a JMS protocol to route the result of polling an interface table to a JMS queue hosted by a J2EE application server. For more information, see *Creating an Event Port* on page 154.

- **Channel**

  A channel represents configured connections to particular instances of back-end or other types of systems. A channel binds one or more event ports to a particular listener managed by an adapter. For more information, see *Creating a Channel* on page 166.

You can employ several techniques when listening for Oracle E-Business Suite events, depending upon your requirements. For information about these techniques, see *Choosing a Listening Technique* on page 174.

Creating an Event Port

The following procedures describe how to create an event port using iWay Explorer.

When you use iWay Explorer with an iWay Business Services Provider (iBSP) implementation, the following port dispositions are available:

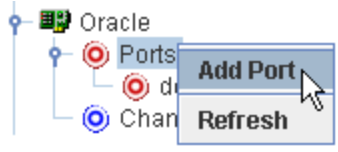
- **File.** The File disposition uses a file URL to specify the destination file name or directory where the event document will be written. During run time, the destination file name may require indexing to avoid overwriting. For more information, see *How to Create an Event Port for File* on page 154.
- **iBSE.** The iBSE disposition enables an event to launch a business service method. For more information, see *How to Create an Event Port for iBSE* on page 156.
- **MSMQ.** The Microsoft Message Queue disposition supports public and private queues. For more information, see *How to Create an Event Port for MSMQ* on page 158.
- **JMSQ.** The JMSQ disposition allows an event to be enqueued to a JMS queue. For more information, see *How to Create an Event Port for JMSQ* on page 159.
- **SOAP.** The SOAP disposition allows an event to launch a business service specified by a WSDL file. A SOAP action is optional; "" is the default value. For more information, see *How to Create an Event Port for SOAP* on page 160.
- **HTTP.** The HTTP disposition uses an HTTP URL to specify an HTTP end point to which the event document is posted. For more information, see *How to Create an Event Port for HTTP* on page 163.
- **MQSeries.** The MQSeries disposition enables an event to be enqueued to an MQSeries queue. Both queue manager and queue name may be specified. For more information, see *How to Create an Event Port for MQSeries* on page 163.

Procedure: How to Create an Event Port for File

To create an event port for File:

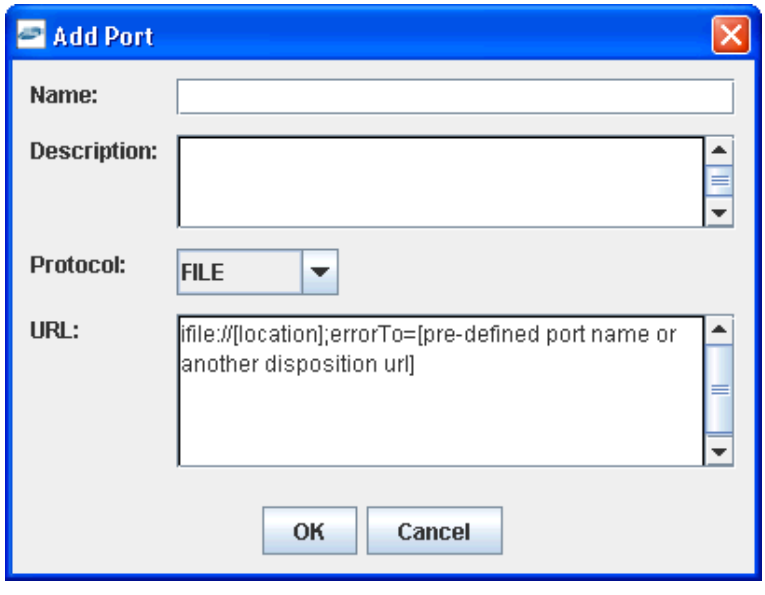
1. Click the *iWay Events* node.
2. In the left pane, expand the *Oracle* node.

The following image shows the options (Add Port and Refresh) that are available in the left pane when you right-click the Ports node.



3. Right-click the *Ports* node and select *Add Port* from the menu.

   The following image shows the Add Port dialog box that opens, where you supply information about the port.



   a. In the Name field, type a name for the port.

   b. In the Description field, type a brief description.

   c. From the Protocol drop-down list, select *FILE*.

   d. In the URL field, type a File destination to which event data is written.

      When pointing iWay Explorer to an iBSE deployment, specify the destination file using the following format:

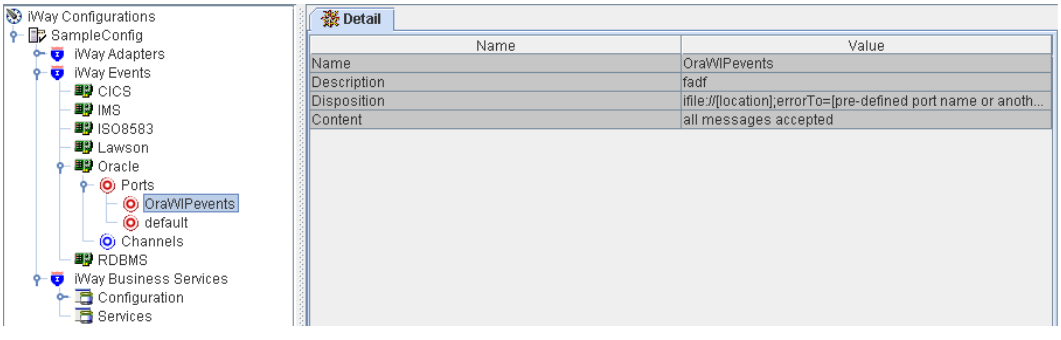
```
ifile://[location];errorTo=[pre-defined port name or another
disposition url]
```

The following table lists and defines the parameters for the disposition.

Parameter	Description
location	Destination and file name of the document where event data is written, for example, `ifile://D:\in\x.txt;errorTo=ifile://D:\error`
errorDest	Predefined port name or another disposition URL where error logs are sent.

4. Click *OK*.

   In the left pane, the event port appears under the ports node. In the right pane, summary information associated with the event port you created appears. In this example, the event port is named OraWIPevents.



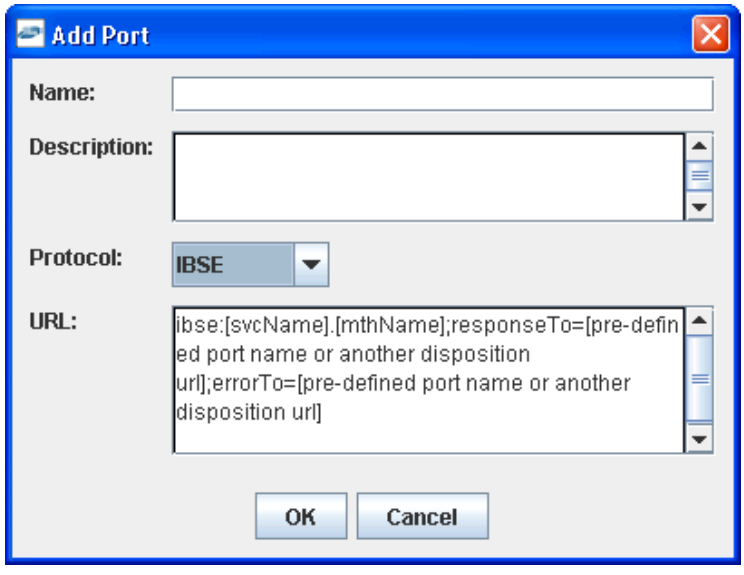
You are ready to associate the event port with a channel. For more information, see *Creating a Channel* on page 166.

Procedure: How to Create an Event Port for iBSE

To create an event port for iBSE:

1. In the left pane, click the *iWay Events* node.
2. Expand the *Oracle* node.
3. Right-click the *Ports* node and select *Add Port* from the menu.

The Add Port dialog box opens, as shown in the following image.



a. In the Name field, type a name for the port.

b. In the Description field, type a brief description.

c. From the Protocol drop-down list, select *iBSE*.

d. In the URL field, enter an iBSE destination in the form of:

```
ibse:svcName.mthName;responseTo=[pre-defined port name or another
 disposition url]; errorTo=[pre-defined port name or another
 disposition url]
```

The following table lists and defines the parameters for the disposition.

Parameter	Description
svcName	Name of the service created with iBSE.
mthName	Name of the method created for the business service.
response To	Location where responses to the business service are posted. Predefined port name or another full URL. Optional.

Parameter	Description
errorTo	Location where error documents are sent. Predefined port name or another full URL. Optional.

4. Click *OK*.

   The port appears under the Ports node in the left pane. In the right pane, a table appears that summarizes the information associated with the event port you created.

You can now associate the event port with a channel. For more information, see *Creating a Channel* on page 166.

Procedure: How to Create an Event Port for MSMQ

To create an event port for MSMQ queue using iWay Explorer:

1. Click the *iWay Events* node.
2. In the left pane, expand the *Oracle* node.
3. Right-click the *Ports* node and select *Add Port* from the menu.

   The Add Port dialog box opens.

   a. In the Name field, type a name for the port.

   b. In the Description field, type a brief description.

   c. From the Protocol drop-down list, select *MSMQ*.

   d. In the URL field, enter an MSMQ destination in the form of:

```
msmq:/host/private$/qName;errorTo=[pre-defined port name or
another disposition url]
```

   The following table lists and defines the parameters for the disposition.

Parameter	Description
host	Machine name where the Microsoft Queuing system is running.
queueType	For private queues, enter *Private$*. Private queues are queues that are not published in Active Directory. They appear only on the local computer that contains them. Private queues are accessible only by Message Queuing applications that recognize the full path name or format name of the queue.

Parameter	Description
queueName	Name of the private queue where messages are placed.
errorTo	Location where error documents are sent. Predefined port name or another full URL. Optional.

4. Click *OK*.

   The port appears under the Ports node in the left pane. In the right pane, a table appears that summarizes the information associated with the port you created.

You are ready to associate the event port with a channel. For more information, see *Creating a Channel* on page 166.

Procedure: How to Create an Event Port for JMSQ

To create an event port for a JMS queue using iWay Explorer:

1. Click the *iWay Events* node.
2. In the left pane, expand the *Oracle* node.
3. Right-click the *Ports* node and select *Add Port* from the menu.

   The Add Port dialog box opens.

   a. In the Name field, type a name for the port.

   b. In the Description field, type a brief description.

   c. From the Protocol drop-down list, select *JMSQ*.

   d. In the URL field, enter a JMS destination.

      When pointing iWay Explorer to an iBSE deployment, use the following format:

```
jmsq:myQueueName@myQueueFac;jndiurl=[myurl];jndifactory=
[myfactory];user=[user];password=[xxx];errorTo=[pre-defined port
name or another disposition url]
```

      The following table lists and defines parameters for the disposition.

Parameter	Description
queue	Name of a queue to which events are emitted.
Connection Factory	A resource that contains information about the JMS Server.

Parameter	Description
jndi_url	The URL to use to contact the JNDI provider. The syntax of this URL depends on which JNDI provider is used. This value corresponds to the standard JNDI property, `java.naming.provider.url`
jndi_factory	Is JNDI context.INITIAL_CONTEXT_FACTORY and is provided by the JNDI service provider.
user	User ID associated with this queue.
password	Password for the user ID.
errorTo	Location to which error logs are sent. Optional. Predefined port name or another disposition URL. The URL must be complete, including the protocol.

4. Click *OK*.

   The port appears under the Ports node in the left pane. In the right pane, a table appears that summarizes the information associated with the port you created.

You can now associate the event port with a channel. For more information, see *Creating a Channel* on page 166.

Procedure: How to Create an Event Port for SOAP

To create an event port for a SOAP disposition:

1. Click the *iWay Events* node.
2. In the left pane, expand the *Oracle* node.
3. Right-click the *Ports* node and select *Add Port* from the menu.

   The Add Port dialog box opens.

   a. In the Name field, type a name for the port.
   b. In the Description field, type a brief description.
   c. From the Protocol drop-down list, select *SOAP*.
   d. In the URL field, enter an SOAP destination, using the following format:

```
soap:[wsdl-url];soapaction=[myaction];method=[web service
method];namespace=[namespace];responseTo=[pre-defined port name or
another disposition URL];errorTo=[pre-defined port name or another
disposition url]
```

The following table lists and defines the parameters for the disposition.

Parameter	Description
wsdl-url	The URL to the WSDL file that is required to create the SOAP message, for example `http://localhost:7001/ibsp/IBSPServlet/test/`*`webservice`*`.ibs?wsdl` where: *`webservice`* Is the name of the web service you created using iWay Explorer. To find this value, navigate to the iWay Business Services tab, expand the *Services* node, select the service you created, and click *Service Description* on the right. The WSDL URL appears in the Address field of the window that opens. You can also open the WSDL file in a third party XML editor (for example, XMLSPY) and view the SOAP request settings to find this value.

Parameter	Description
soapaction	The method called by the SOAP disposition. For example: *webservice.method@test@@* where: *webservice* Is the name of the web service you created using iWay Explorer. *method* Is the method being used. *test* Is the license that is being used by the web service. This value can be found by navigating to the iWay Business Services tab and opening the *Service Description* link in a new window. Perform a search for *soapAction*. You can also open the WSDL file in a third party XML editor (for example, XMLSPY) and view the SOAP request settings to find this value.
method	The web service method you are using. This value is found in the WSDL file.
namespace	The XML namespace you are using. This value is found in the WSDL file.
responseTo	The location to which responses are posted, which can be a predefined port name or another URL. Optional.
errorTo	Location to which error logs are posted which can be a predefined port name or another URL. Optional.

4. Click *OK*.

   The event port appears under the Ports node in the left pane. In the right pane, a table appears that summarizes the information associated with the evnet port you created.

You are ready to associate the event port with a channel. For more information, see *Creating a Channel* on page 166.

Procedure: How to Create an Event Port for HTTP

To create an event port for an HTTP disposition using iWay Explorer:

1. Click the *iWay Events* tab.
2. In the left pane, expand the *Oracle* node.
3. Right-click the *Ports* node and select *Add Port* from the menu.

   The Add Port dialog box opens.

   To connect iWay Explorer to an iBSE deployment, follow the steps in *How to Create an Event Port for iBSE* on page 156.

   The following table lists and describes the disposition parameters for HTTP.

Parameter	Description
url	URL target for the post operation.
respDest	Location where responses are posted. Predefined port name or another full URL. Optional. Predefined port name or another disposition URL. The URL must be complete, including the protocol.
host	Name of the host on which the web server resides.
port	Port number on which the web server is listening.
uri	Universal resource identifier that completes the url specification.

You can now associate the event port with a channel. For more information, see *Creating a Channel* on page 166.

Procedure: How to Create an Event Port for MQSeries

To create an event port for an MQSeries queue using iWay Explorer:

1. Click the *iWay Events* node.
2. In the left pane, expand the *Oracle* node.
3. Right-click the *Ports* node and select *Add Port* from the menu.

The Add Port dialog box opens.

a. In the Name field, type a name for the port.

b. In the Description field, type a brief description.

c. From the Protocol drop-down list, select *MQSeries*.

d. In the URL field, enter an MQSeries destination.

When pointing iWay Explorer to an iBSE deployment, use the following format:

```
mqseries:/qManager/qName;host=[hostName];
port=[port];channel=[channelname];
errorTo=[predefined port name or another disposition url]
```

The following table lists and defines the disposition parameters for MQSeries.

Parameter	Description
qManager	Name of the queue manager to which the server must connect.
qName or respqueue	Name of the queue where messages are placed.
host	Name of the host on which MQSeries resides (MQ client only).
port	Number to connect to an MQ server queue manager (MQ client only).
channel	Case-sensitive name of the channel that connects with the remote MQ Server queue manager (MQ client only). The default channel name is SYSTEM.DEF.SVRCONN.
errorTo	Location where error documents are sent. Predefined port name or another full URL. Optional.

4. Click *OK*.

The port appears under the Ports node in the left pane. In the right pane, a table appears that summarizes the information associated with the even port you created.

You can now associate the event port with a channel. For more information, see *Creating a Channel* on page 166.

Editing or Deleting an Event Port

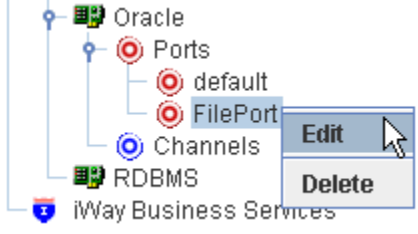
The following procedures describe how to edit or delete an event port using iWay Explorer.

Procedure: How to Edit an Event Port

To edit an existing event port using iWay Explorer:

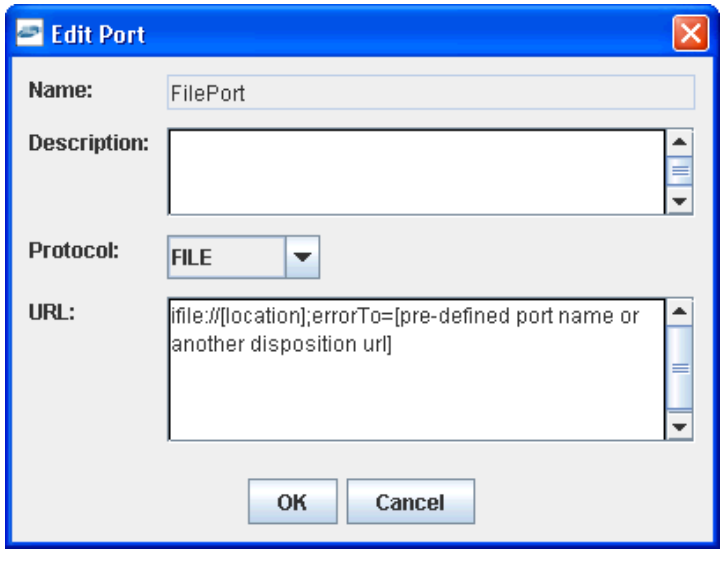
1. Right-click the event port you want to edit.

   The following image shows the options (Edit and Delete) that appear in the left pane when you right-click the event port. In this example, the event port is named FilePort.



2. Select *Edit* from the menu.

   The following image shows a sample Edit Port dialog box that opens.



3. Make the required changes to the event port configuration fields.
4. Click *OK*.

Procedure: How to Delete an Event Port

To delete an existing event port using iWay Explorer:

1. Right-click the event port you want to delete.
2. Select *Delete* from the menu.
3. The event port disappears from the list in the left pane.

Using the Default Event Port

When using iWay Explorer to connect to Oracle E-Business Suite and listen for events, a default event port is available at all times as shown in the following image.

The default event port can be used for testing purposes or when you do not want to route event data to a specific port you configured. The default port is enabled when you start a channel that does not have a specific event port assigned.

The default event data is actually a file disposition that writes to an out.xml file in the following output directory:

```
ifile://./eventOut/out.xml
```

Creating a Channel

The following procedure describes how to use iWay Explorer (Java Swing) to create a channel for an Oracle E-Business Suite event. All defined event ports must be associated with a channel.

Procedure: How to Create a Channel

To create a channel using iWay Explorer:

1. In the left pane, expand the *iWay Events* node below the configuration you created, for example, SampleConfig.

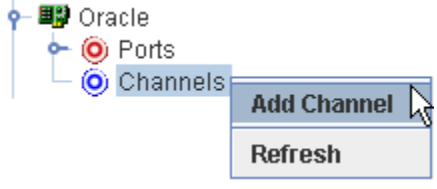
   The list of adapters appears.

2. Expand the adapter node, for example, Oracle.

The following image shows the Ports and Channels nodes that appear under the Oracle node.

3. Right-click *Channels*.

   The following image shows the option (Add Channel) that is available when you right-click Channels.



4. Select *Add Channel*.

The following image shows the Add Channel dialog box that opens, where you supply information about the channel.

a. In the Name field, type a name for the channel, for example, OracleChannel.

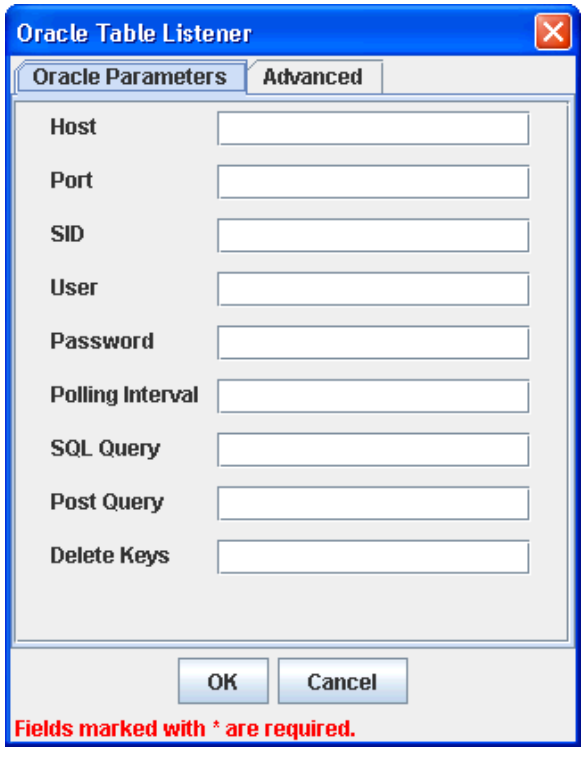
b. In the Description field, type a brief description.

c. From the Protocol drop-down list, select a server that supports event handling.

d. From the Available Ports area, select a port you want to assign to this channel and click the right arrow to move it to the Selected Ports area.

5. Click *Next*.

The following image shows a sample Oracle Table Listener dialog box that opens, where you supply information about the table listener and provide parameters.



6. Enter the values according to the information in the following table.

   The following table lists and describes the required parameters for the JDBC Thin Driver.

Field	Description
Host	Name of the server on which the Oracle E-Business Suite database instance resides.
Port	Port number on which the database is listening.
SID	Unique name of the database service, chosen by the database administrator or the person who installed Oracle E-Business Suite.

Field	Description
User	Oracle database user ID to access the Oracle database underlying the Oracle E-Business Suite system. The user ID must have database access to the interface tables being accessed.
Password	Password associated with the specified user ID.
Polling Interval	Interval, in milliseconds, at which to check for new input.

SQL Query	SQL SELECT statement that the listener issues to poll the table.
	
	If the SQL statement includes a date column or long text column, you must provide a value for the SQL Post-query parameter. The value you provide must not contain a date column or a long text column. This applies whether you provide an SQL statement here or rely upon the default.
	
	For example, the following SELECT statement retrieves all unprocessed records from the DISCRETE_JOBS table:
	
	`SELECT * FROM WIP_DISCRETE_JOBS D`
	`WHERE DJ.WIP_ENTITY_ID >`
	`(SELECT WIP_ENTITY_ID`
	`FROM WIP.TEMP_NEW_WORK_ORDER_ENTITY_ID)`
	
	Important: When a SQL Query joins two or more tables, a SQL Post Query must be used. Also, do not use a semicolon at the end of a SQL statement for a SQL Query or a SQL Post Query.

Field	Description
Post Query	A SQL statement that is executed after each new record is read from the table. Case sensitive: the case used to specify the column names must match the case used in the SELECT statement that polled the table. If the SQL Query property was omitted so that a default SELECT statement polled the table, the case used to specify the column names must match the case used to define the columns in the DBMS native schema. If you do not specify a value for SQL Post-query, each record read from the table is deleted after it is read. How this happens depends on whether you specify the Delete Keys property. If you: **Specify the Delete Keys property**, by default the adapter issues a DELETE statement with a WHERE clause containing every key column specified for the Delete Keys property. At run time this is faster than if you had not specified the Delete Keys property if there is an index on the key or if there are fewer key columns than there are columns in the SELECT statement that polled the table. **Do not specify the Delete Keys property**, by default the adapter issues a DELETE statement with a WHERE clause that specifies every column from the SELECT statement that polled the table. You can choose to retain the table data after it is read by specifying a value for this parameter, as shown in the examples that follow. **Note:** The SQL Post-query and Delete Keys parameters are mutually exclusive, because Delete Keys applies to the default DELETE statement, and SQL Post-query overrides the default DELETE statement. You can provide a value for one or the other, but not for both. There are two field operators, ? and ^, that you can use in a post-query SQL statement. **Important:** When a SQL Query joins two or more tables, a SQL Post Query must be used. Also, do not use a semicolon at the end of a SQL statement for a SQL Query or a SQL Post Query.

Field	Description
Delete Keys	Comma-separated list of key columns to be used in the default DELETE statement. DELETE operates on keys, so specify the table key columns. This is case sensitive: the case used to specify the column names must match the case used in the SELECT statement that polled the table. If the SQL Query property was omitted so that a default SELECT statement polled the table, the case used to specify the column names must match the case used to define the columns in the DBMS native schema. **Note:** The Delete Keys and SQL Post Query parameters are mutually exclusive, because Delete Keys applies to the default DELETE statement, and SQL Post Query overrides the default DELETE statement. You can provide a value for one or the other, but not for both. For more information, see the description of the SQL Post-query parameter in this table.

7. After you provide the required information, click *OK*.

   The following image shows the channel you created beneath the Channels node in the left pane. In this example, the channel is named OracleChannel.

   In the right pane, a table summarizes the information associated with the channel.

   You are ready to start the channel to listen for events.

8. In the left pane, right-click the channel, for example, OracleChannel, and select *Start*.

   The channel you created is now active.

   a. To stop the channel at any time, right-click the channel.

   b. Select *Stop*.

Procedure: How to Edit a Channel

To edit an existing channel:

1. In the left pane, right-click the channel you want to edit, for example, OracleChannel.
2. Select *Edit*.

The Edit Channel dialog opens.

3. Make the required changes to the channel configuration and click OK.

Procedure: How to Delete a Channel

To delete an existing channel:

1. In the left pane, right-click the channel you want to delete, for example, OracleChannel.
2. Select *Delete*.
   The channel disappears from the list in the left pane.

The Post Query Parameter Operators

You can use two special field operators, ? and ^, with the Post Query parameter. Both of these operators dynamically substitute database values in the SQL post-query statement at run time.

- `?fieldname` is evaluated at run time as `field = value`

  The ? operator is useful in UPDATE statements:

  ```
  UPDATE table WHERE ?field
  ```

  For example, the following statement

  ```
  UPDATE Stock_Prices_Temp WHERE ?RIC
  ```

  might be evaluated at run time as:

  ```
  UPDATE Stock_Prices_Temp WHERE RIC = 'PG'
  ```

- `^fieldname` is evaluated at run time as `value`

  The ^ operator is useful in INSERT statements:

  ```
  INSERT INTO table VALUES (^field1, ^field2, ^field3, ...)
  ```

  For example, the following statement

  ```
  INSERT INTO Stock_Prices_Temp VALUES (^RIC, ^Price, ^Updated)
  ```

  might be evaluated at run time as:

  ```
  INSERT INTO Stock_Prices_Temp VALUES ('PG', 88.62, '2003-03-18
  16:24:00.0')
  ```

Choosing a Listening Technique

You can detect an Oracle E-Business Suite event by using an RDBMS Table Listener for Oracle E-Business Suite. The Table Listener polling technology enables you to specify SQL SELECT statements to execute periodically. After data is polled, it passes through the event port for additional processing.

You can poll a relational or non-relational database directly and send the results to a file or JMS message queue. You use the following techniques to listen to an Oracle E-Business Suite event:

- Standard event processing with row tracking

  The listener polls a table, sends each newly inserted row to a destination you specify (known as the disposition), and uses a control table to keep track of the row that was most recently read. The control table prevents the most recently read row from being reread during the next listening cycle.

  You can apply this flexible yet simple technique in most situations. For more information, see *Standard Event Processing With Row Tracking* on page 175.

- Standard event processing with row removal

  The listener polls a table, sends each newly inserted row to a destination you specify, and then deletes the new row from the table to prevent it from being reread during the next listening cycle.

  You can apply this technique when the source table is being used to pass data to the adapter, and the table rows do not need to persist. Rows are deleted as they are processed. For more information, see *Standard Event Processing With Row Removal* on page 178.

- Trigger-based event processing

  At design time you assign triggers to a joined group of tables. At run time the triggers write information about table changes to a common control table. The listener polls the control table and sends information about the table changes to a destination you specify. The listener deletes new rows from the control table to prevent them from being reread during the next listening cycle.

  You can apply this technique when listening to events in a group of large joined tables, or when you need to know if a row has been updated or deleted. For more information, see *Trigger-based Event Processing* on page 180.

Configuration requirement: Copy the JDBC driver libraries if you have not yet done so to the iway7\lib directory.

Standard Event Processing With Row Tracking

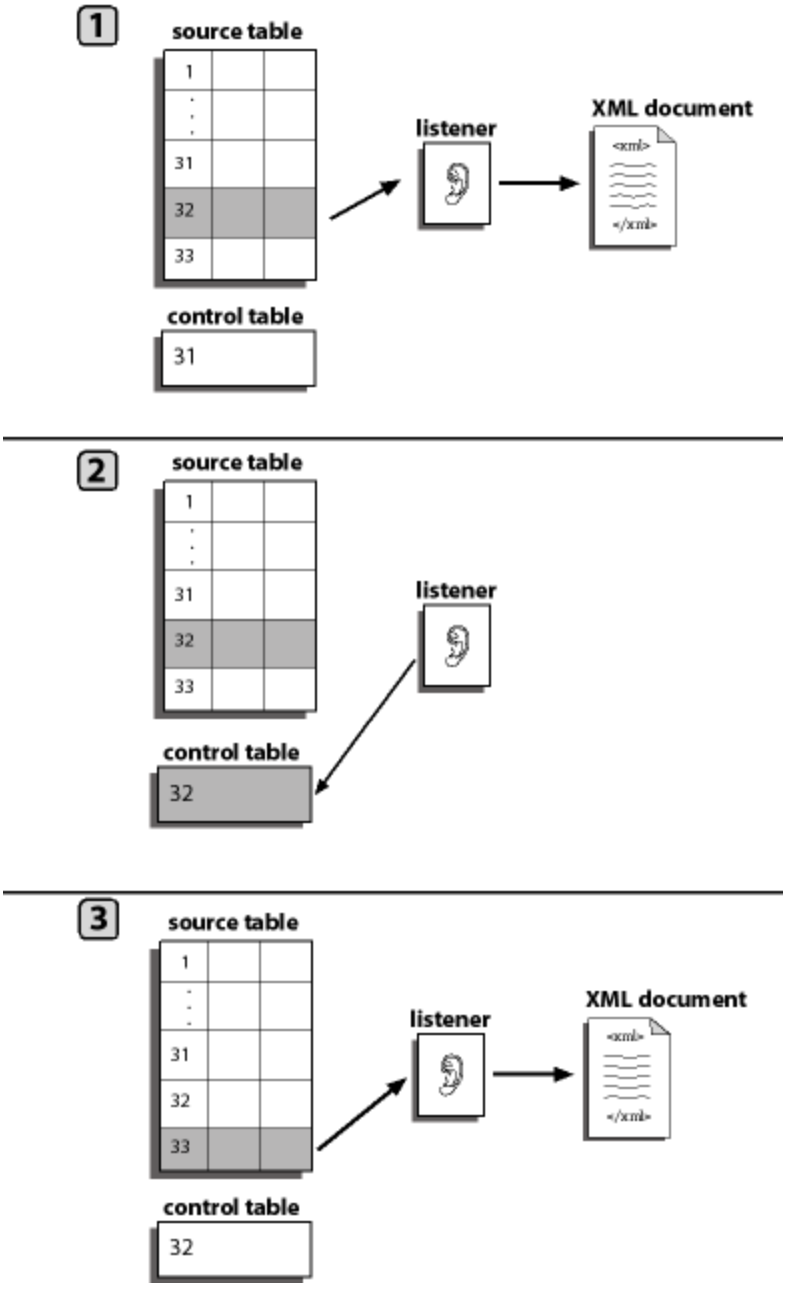
The standard event processing with row tracking technique enables you to listen to the source table without removing its rows. It requires you to create a single-cell control table that keeps track of the last new row the RDBMS Table Listener for Oracle E-Business Suite read from the source table.

The control table's one column corresponds to a column (or to a group of columns) in the source table that is unique, sortable, and indicates how recently the row was added to the source table relative to the other rows. For example, the first row added to the source table has the lowest value, and the last row added has the highest value. This value is the "event key."

When you create the control table, initialize it to the event key of the row most recently added to the source table. When you specify the listener's properties, configure the listener's Post Query property to automatically update the event key of the control table.

Each time the listener queries the source table, it looks for rows added since the last query—that is, for rows whose event key is greater than the current value of the field in the control table. It reads each row of this type and returns it to the specified destination using an XML document. To ensure that the row is not read again the next time the listener queries the table, the listener updates the field in the control table to match the value of the row just read from the source table.

The following figure illustrates standard event processing with row tracking. It shows the source and control tables, the listener, and the XML document at various stages.



In the previous figure:

1. The Table Listener queries the source table and copies each source table row whose event key is greater than the event key of the control table. The listener copies the row to an XML document and sends it to the destination defined in the port disposition.
2. The listener updates the event key in the control table to match the row it has most recently read.
3. The listener copies the next source table row to an XML document.

The process repeats.

To implement this event processing technique, see *How to Implement Standard Event Processing With Row Tracking* on page 177.

Procedure: How to Implement Standard Event Processing With Row Tracking

To implement standard event processing with row tracking:

1. Create a control table.
2. Configure an RDBMS Table Listener for Oracle E-Business Suite using iWay Explorer or iWay web console.

   In addition to the required listener properties, for standard event processing with row tracking you must also provide values for the following optional properties:

   SQL Query, the SQL SELECT statement that identifies the source table to which the adapter listens, and with which it queries the table.

   Post Query, the SQL statements that maintain the field in the control table.

For instructions for configuring a listener, see *Creating a Channel* on page 166.

For an example of creating the control table, see the following topic.

Example: Creating the Control Table for an Oracle E-Business Suite Event

Follow the steps in this example to create a table named TEMP_NEW_YORK_ORDER_ENTITY that has a single field named WIP_ENTITY_ID. You specify this table when you configure the RDBMS Table Listener for Oracle E-Business Suite, as described in *Creating a Channel* on page 166.

When discrete jobs are created through the Oracle E-Business Suite graphical interface, an entry is created in the WIP.WIP_DISCRETE_JOBS table. For this example, you configure an event to detect new entries to this table. You use the standard event processing with row tracking technique. (Oracle E-Business Suite processing cannot delete rows from the table.) To accomplish this, first create a simple table to track of the records processed.

1. From within Oracle SQL*PLUS, run the following SQL:

   ```
   CREATE TABLE WIP.TEMP_NEW_WORK_ORDER_ENTITY_ID
   (
      WIP_ENTITY_ID   NUMBER
   )
   ```

   This creates a single table with a single field.

   Note: Oracle SQL*Plus is part of the Oracle client software. If it is not installed, contact your Oracle Database Administrator.

   You must be logged in under the APPS schema or a similar ID that has access rights to the Oracle E-Business Suite WIP schema.

2. Create a single record in this table and seed it with the highest WIP_ENTITY_ID ID from your system. You can obtain this from the WIP.WIP_DISCRETE_JOBS table.

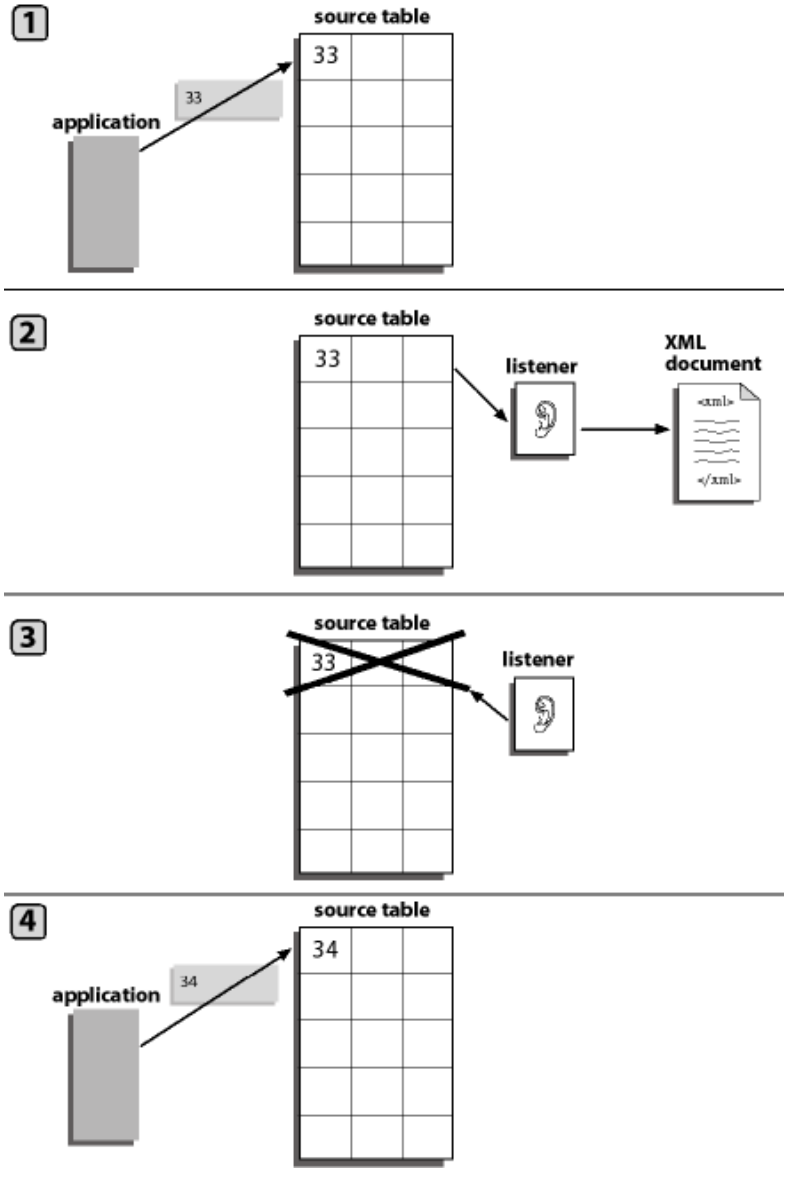
   This sets the value at which to start detecting events as records enter the WIP_DISCRETE_JOBS table.

After you create a simple table in Oracle, you must configure the Table Listener, as described in *Creating a Channel* on page 166.

Standard Event Processing With Row Removal

The standard event processing with row removal technique assumes that the source table is being used as a conduit to pass the data to the adapter, and that the table rows do not need to persist. The RDBMS Table Listener for Oracle E-Business Suite periodically queries the source table. When it finds a row, it reads it and returns it to the Reply_to destination via an XML document. To ensure that the row is not read again when the Table Listener next queries the table, the listener then deletes the row from the table.

The following figure illustrates standard event processing with row removal. It shows the application, source table, listener, and the XML document at various stages in the event processing.



In the previous figure:

1. Your application inserts a new row into the source table.

2. The listener queries the source table and copies the new row to an XML document and sends it to the destination defined in the port disposition using the File protocol.
3. The listener deletes the source table row to ensure that the row is not read again when the listener next queries the table.
4. The application inserts a new row into the source table.

The process repeats itself.

To implement this event processing technique, see *How to Implement Standard Event Processing With Row Removal* on page 180.

Procedure: How to Implement Standard Event Processing With Row Removal

To implement the standard event processing with row removal technique:

1. Configure an RDBMS Table Listener.
2. In addition to the required listener properties, provide values for the following optional properties:

   SQL Query, the SQL SELECT statement that identifies the source table to which the adapter listens, and with which it queries the table.

   Issue Post Query Delete, which automatically deletes each record after it was read.

For detailed instructions for configuring a listener, see *Creating a Channel* on page 166. For information on SQL post query parameters, see *The Post Query Parameter Operators* on page 173.

Trigger-based Event Processing

Trigger-based event processing is a technique for listening to multiple joined Oracle E-Business Suite tables. It is also helpful for detecting when a row was deleted or updated.

The trigger-based technique provides the following benefits:

- Improved performance when listening to events in a group of large joined tables

  When processing joined tables, Oracle creates a Cartesian product working table. When the joined tables are large, the interim working table is very large. The standard technique of processing Oracle E-Business Suite events, in which the adapter periodically listens to the entire structure of joined tables, can consume a significant amount of computing resources.

  The trigger-based technique avoids this overhead by requiring the RDBMS Table Listener for Oracle E-Business Suite to query a single small control table and by writing to the control table only when an event actually occurs.

- Increased number of event types that the adapter recognizes

  Using the trigger-based technique, you can tell when a row was updated, deleted, or inserted. Using the standard technique, you can tell only when a row was inserted.

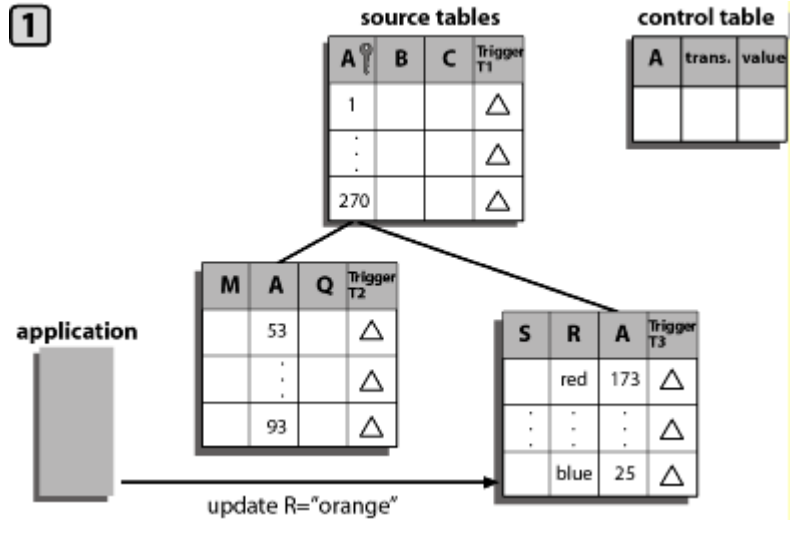
To use the trigger-based technique, you assign a trigger to each table that you want to monitor. When a value changes, it fires the corresponding trigger, which writes data to a control table. The iWay Application Adapter for Oracle E-Business Suite listens to this control table by running a query against it. When it finds a row in the control table, it reads it and returns it to the port disposition created when the port is configured via an XML document. To ensure that the row is not read again when the listener next queries the table, the listener then deletes the row from the table.

The trigger-based technique enables you to recognize changes to an entity. For the purposes of this discussion, an entity is a real-world object that is represented in the database by a hierarchical set of tables.

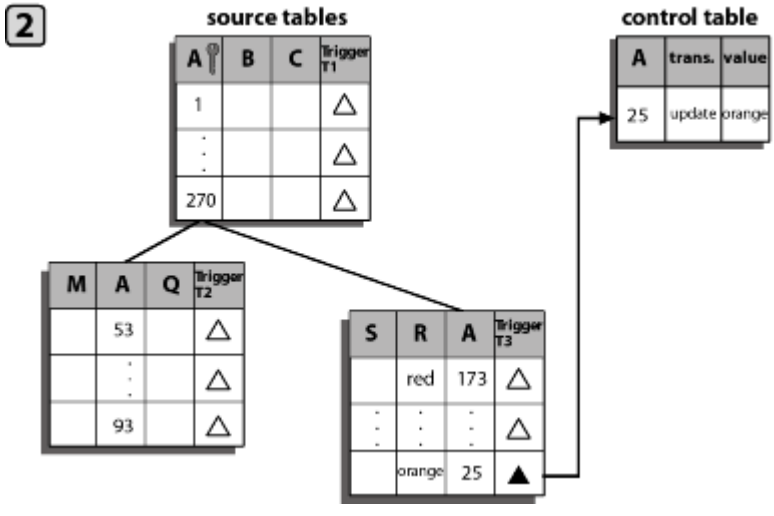
You manage the triggers using SQL*Plus or a similar tool and configure the event using iWay Explorer.

The following five figures illustrate the steps involved in trigger-based event processing. They show the source and control tables, the listener, the application, and the XML document at various stages in the event processing.

1. Your application updates a row in a group of related source tables.

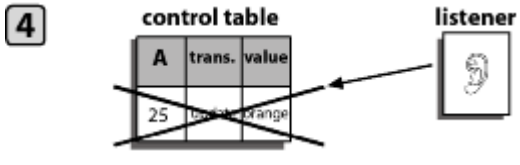


2. The update causes a row trigger to fire in the changed table. The trigger inserts a row into the control table. The new control table row includes the key value (25), the type of transaction (update), and the new cell value (orange).

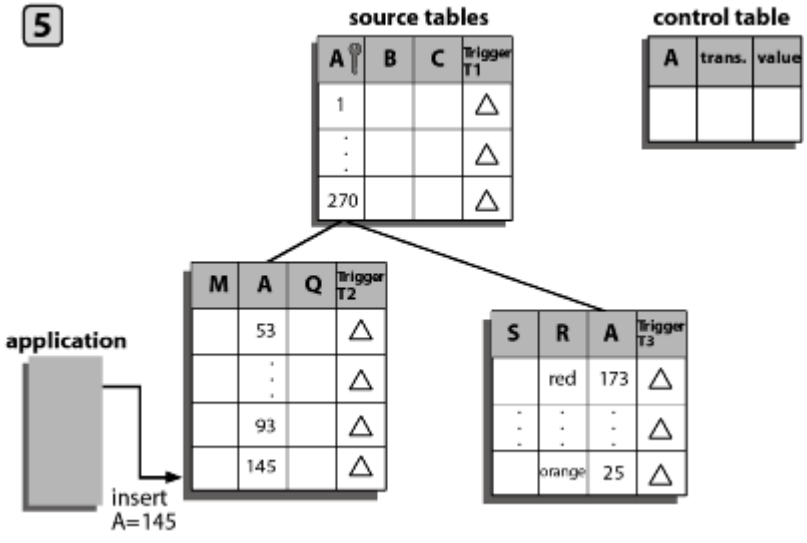


3. The listener queries the control table and copies the new row to an XML document. It sends the document to the destination defined in the port disposition.

4. The listener deletes the control table row to ensure that the row is not read again when the listener next queries the table.



5. The application inserts a new row into one of the source tables.



The process repeats itself.

For a summary of how to implement this technique, see *How to Implement Trigger-based Event Processing* on page 183.

Procedure: How to Implement Trigger-based Event Processing

To implement the trigger-based event processing technique:

1. Create the control table.

   The purpose of the control table is to capture the key of each entity that changed, regardless of which of the entity tables changed.

   You can store a variety of information in the control table, including the key of the entity that was inserted, updated, or deleted, and the name of the table and field that was updated.

   The design of the control table is a function of the business logic of your application. For example, you can choose between creating one control table for a group of joined source tables or one control table per source table. Among the issues to consider are the kinds of events to monitor (insertions, deletions, and/or updates), and whether you want to monitor only the highest-level table in a group of joined tables or all of the tables in the group.

2. Assign triggers to the source tables.

The triggers you assign, and to which tables you assign them, is determined by what kind of change you want to monitor. The triggers implement much of the event-processing logic.

For example, consider a bill of material scenario. (A bill of materials is a list of all the parts required to manufacture an item, the subparts required for the parts, and so on. The complete item/parts/subparts relationship can extend to several levels, creating a data structure like a tree with the finished item as the root.) In a bill of materials, where each level in the parts hierarchy is represented by a separate table, you might assign a trigger to only the highest-level table (the finished product), or you might assign triggers to all tables (the finished product and its parts and subparts).

As another example, if multiple changes are made to the same row during one listener cycle, you could configure the event adapter to record all the changes. If a row was inserted and then updated, both changes would be logged.

3. Configure the RDBMS Table Listener for Oracle E-Business Suite when creating a channel using iWay Explorer.

   In addition to the required listener properties, for trigger-based event processing you also must provide values for the following optional properties:

   SQL Query, the SQL SELECT statement that identifies the control table to which the adapter listens, and with which it queries the table to determine changes in the source tables.

   Post Query, to identify the rows that the adapter automatically deletes from the control table.

For detailed instructions for configuring a listener, see *Creating a Channel* on page 166.

Example: Trigger on WIP_ENTITY_NAME Column

The following trigger fires when a change is made to the WIP_ENTITY_NAME column of the WIP.WIP_ENTITIES table. When it fires, it writes the relevant values to the control table IWAY.IWAY_PO_CDC.

```
CREATE OR REPLACE TRIGGER IWAY.IWAY_PO_CDC_WE_TRG
AFTER INSERT OR DELETE OR UPDATE OF WIP_ENTITY_NAME
ON WIP.WIP_ENTITIES
FOR EACH ROW
BEGIN
IF INSERTING THEN
   INSERT INTO IWAY.IWAY_PO_CDC
      VALUES (
         :NEW.WIP_ENTITY_ID,
         :NEW.ORGANIZATION_ID,
             'UPDATE');
ELSE
   INSERT INTO IWAY.IWAY_PO_CDC
      VALUES (
         :OLD.WIP_ENTITY_ID,
         :OLD.ORGANIZATION_ID,
             'UPDATE');
END IF;
EXCEPTION
   WHEN DUP_VAL_ON_INDEX THEN
      NULL;         -- Record already exists
END;
```

Chapter 11

Oracle E-Business Suite Troubleshooting and Error Messages

The following topics explain the limitations and workarounds when connecting to Oracle E-Business Suite.

The adapter-specific errors listed in this section can arise if you are using the adapter with an iBSP configuration.

In this chapter:

Oracle E-Business Suite Troubleshooting

This topic provides troubleshooting information for iWay Application Adapter for Oracle E-Business Suite for the following categories:

- iWay Explorer
- iBSP

Reference: **Error Messages in iWay Explorer**

The following table lists and describes errors and corresponding solutions for iWay Explorer.

Error	Solution
Cannot connect to the iWay Application Adapter for Oracle E-Business Suite from iWay Explorer.	Ensure that: ❑ Oracle E-Business Suite is running. ❑ The Oracle E-Business Suite user ID and password are correct. ❑ The port number is correct.

Error	Solution
Cannot connect to the Oracle E-Business Suite target through iWay Explorer and a login error appears.	You have provided invalid connection information for Oracle E-Business Suite or the wrong JAR file is in the lib directory. See the *iWay Installation and Configuration* manual for information on JAR files.
Oracle does not appear in the iWay Explorer Adapter node list.	Ensure that the Oracle JAR files are added to the lib directory. See the *iWay Installation and Configuration* manual for information on JAR files.

iWay Business Services Provider Error Messages

This topic discusses the different types of errors that can occur when processing iWay Business Services through iWay Business Services Provider.

General Error Handling in iWay Business Services Provider

iWay Business Services Provider serves as both a SOAP gateway into the adapter framework and as the engine for some of the adapters. In both design time and execution time, various conditions can cause errors in iBSP when web services that use adapters are running. Some of these conditions and resulting errors are exposed the same way, regardless of the specific adapter; others are exposed differently, based on the adapter being used. This topic explains what you can expect when you encounter some of the more common error conditions on an adapter-specific basis.

Usually, the SOAP gateway (*agent*) inside iBSP passes a SOAP request message to the adapter required for the web service. If an error occurs, the way it is exposed depends on the adapter and the API or interfaces that the adapter uses. A few scenarios cause the SOAP gateway to generate a SOAP fault. In general, when the SOAP agent inside iBSP receives an invalid SOAP request, a SOAP fault element is generated in the SOAP response. The SOAP fault element contains fault string and fault code elements. The fault code contains a description of the SOAP agent error.

The following SOAP response document results when iBSP receives an invalid SOAP request.

```
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

 <SOAP-ENV:Body>
      <SOAP-ENV:Fault>
         <faultcode>SOAP-ENV:Client</faultcode>
         <faultstring>Parameter node is missing</faultstring>
      </SOAP-ENV:Fault>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

In the previous example, iBSP did not receive an element in the SOAP request message that is mandatory for the WSDL for this web service.

Adapter-Specific Error Handling

When an adapter raises an exception during execution, the SOAP agent in iBSP produces a SOAP fault element in the generated SOAP response. The SOAP fault element contains fault code and fault string elements. The fault string contains the native error description from the adapter target system. Because adapters use the target system interfaces and APIs, whether an exception is raised depends on how the target systems interface or API treats the error condition. If a SOAP request message is passed to an adapter by the SOAP agent in iBSP, and that request is invalid based on the WSDL for that service, the adapter may raise an exception yielding a SOAP fault.

Although it is almost impossible to anticipate every error condition that an adapter may encounter, the following examples describe how adapters handle common error conditions and how they are then exposed to the web services consumer application.

Example: iWay Application Adapter for Oracle E-Business Suite Invalid SOAP Request

When the adapter receives a SOAP request message that does not conform to the WSDL for the web service being executed, the following SOAP response is generated:

```
<SOAP-ENV:Envelope xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"

xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
   <m:CARRIERResponse xmlns:m="urn:schemas-iwaysoftware-com:iwse"
   xmlns="urn:schemas-iwaysoftware-com:iwse"
   cid="2A3CB42703EB20203F91951B89F3C5AF">
   <PS8>
      <error>Cannot find Component Interface {VARRIER}
```

```
(91,2)Initialization
    failed (90,7)Not Authorized (90,6)Failed to execute PSSession request
Cannot find Component Interface {VARRIER} (91,2)</error>
        </PS8>
      </m:CARRIERResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Example: Failure to Connect to Oracle E-Business Suite

When the iWay Application Adapter for Oracle E-Business Suite cannot connect to Oracle when executing a web service, the following SOAP response is generated.

```
<?xml version="1.0" enco
ding="ISO-8859-1" ?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
  <SOAP-ENV:Fault>
   <faultcode>SOAP-ENV:Server</faultcode>
  <faultstring>java.lang.Exception: Error Logon to Oracle Applications
   System<faultstring>
 </SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Invalid SOAP Request

When the adapter receives a SOAP request message that does not conform to the WSDL for the web service being executed, the following SOAP response is generated.

```
<?xml version="1.0" encoding="ISO-8859-1"
 ?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
 <SOAP-ENV:Fault>
  <faultcode>SOAP-ENV:Server</faultcode>
  <faultstring>RPC server connection failed: Connection refused:
connect</faultstring>
 </SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Empty Result From a Request

Note: The condition for this adapter does not yield a SOAP fault.

When the adapter executes a SOAP request using input parameters passed that do not match records in the target system, the following SOAP response is generated.

```
<SOAP-ENV:Envelope xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
   <SOAP-ENV:Body>
      <m:RunDBQueryResponse xmlns:m="urn:schemas-iwaysoftware-com:iwse"
        xmlns="urn:schemas-iwaysoftware-com:iwse"
        cid="2A3CB42703EB20203F91951B89F3C5AF">
        <RunDBQueryResult run="1" />
     </m:RunDBQueryResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Appendix A

Configuring the Application Adapter for Oracle E-Business Suite in an iWay Environment

After you successfully configure the adapter to represent a particular adapter target, the adapter can be assigned to an iWay Service Manager channel.

In this appendix:

Configuring the Application Adapter for Oracle E-Business Suite in iWay Service Manager

Before configuring the adapter in iWay Service Manager, you must first create a target, which represents a connection to a backend system, using iWay Explorer. For more information on configuring targets and connections using iWay Explorer, see *Configuring and Managing Connections to Oracle E-Business Suite* on page 47.

You configure the adapter in the iWay Service Manager console. The configuration process creates run-time connection and persistent data files within Service Manager. The configuration process interrogates the Service Manager repository entries that were built when the target and connection were created using iWay Explorer. The define adapter process creates the run-time repository based on the design-time repository.

Procedure: How to Define an Adapter

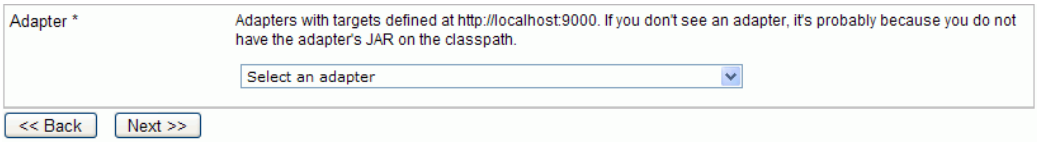
To define an adapter:

1. In the Service Manager console, select *Registry*, then *Adapters*.
2. Click *Add*.

   The iBSP URL pane opens, as shown in the following image.

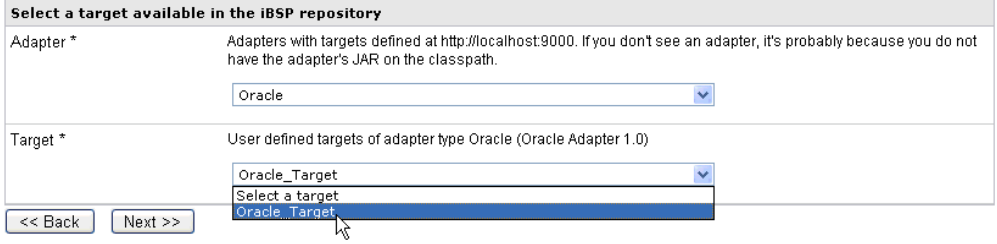
3. Enter your iBSP URL, which is the location of the Service Manager repository, for example, `http://localhost:9000`. This field is required.
4. Click *Next*.

An adapter selection pane opens, as shown in the following image.



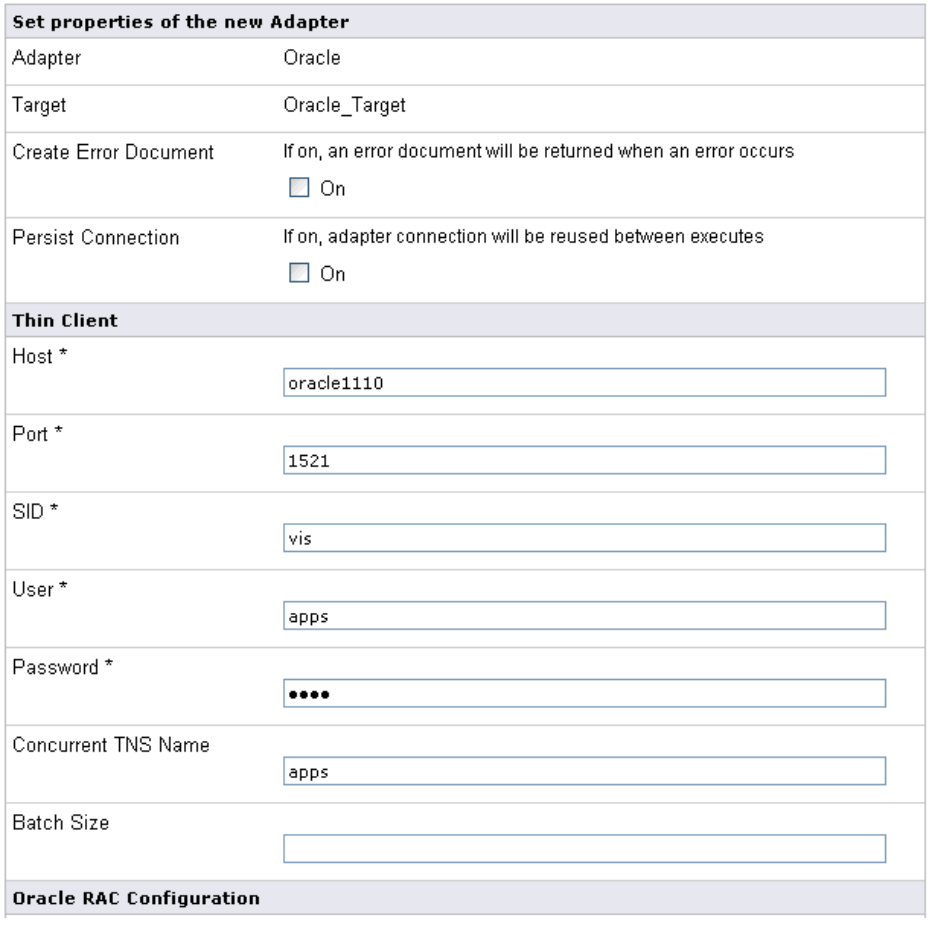
5. From the Adapter drop-down list, select *Oracle*, then click *Next*.

   The Target section is added, as shown in the following image.



6. From the Target drop-down list, select a target you configured for the adapter in iWay Explorer, then click *Next*.

The connection information associated with the target selected is displayed.



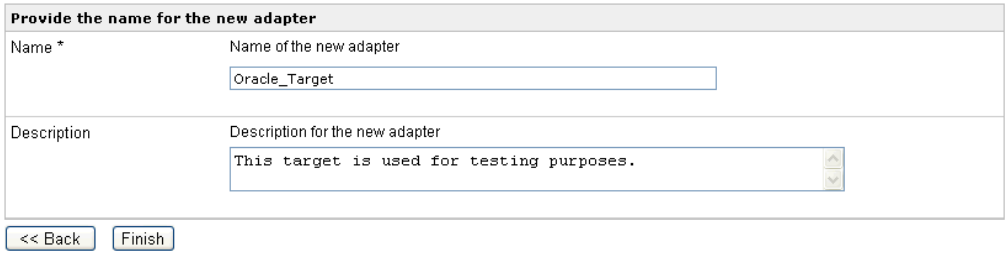
a. Select whether to return an error document when an error occurs.

b. Select whether an adapter connection will be reused between executes.

c. Review the connection information you specified in iWay Explorer. You can change or update any information.

7. Click *Next*.

The adapter name and description pane opens.

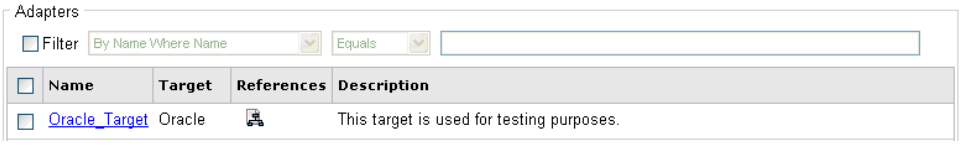


8. Provide a name, for example, Oracle_Target, and optionally, a description for the adapter, and click *Finish*.

   The adapter appears in the adapters list, as shown in the following image.

Procedure: How to Modify or Update an Adapter Connection

The following image shows the Adapters pane which displays the name of the adapter and the description (optional).



To modify or update an adapter connection:

1. From the Adapters list, click the adapter you defined. In this example, click *Oracle_Target*.

   The pane that displays the target connection information opens. You cannot change the name of the adapter or the target, but you can edit the connection information.

2. After you modify the connection information, click *Update Connection Properties*.
3. After you make changes or additions to the adapter target in iWay Explorer, click *Update Adapter Data*.
4. Click *Finish*.

Appendix B

Configuring the Application Adapter for Oracle E-Business Suite in iWay Integration Tools Designer

After you successfully configure the adapter to represent a particular adapter target, the adapter can be used within a process flow.

In this appendix:

Using the Adapter in iWay Integration Tools Designer

You can make an adapter available to a process flow created in iWay Integration Tools (iIT) Designer, a GUI-based tool, used to build stateless process flows that execute within iWay Service Manager (iSM). The adapter can be incorporated as a node, called an Adapter object, in an iWay process flow, allowing you to integrate it easily into a business process solution.

When creating an Adapter object, you can add an adapter to the object in one of two ways:

- Add an adapter that has already been defined either in a process flow or in iWay Service Manager. For more information, see *How to Add a Previously Defined Adapter* on page 197.
- Add an adapter that has *not* been previously defined in iIT Designer or iWay Service Manager. For more information, see *How to Define a New Adapter in iWay Integration Tools Designer* on page 200.

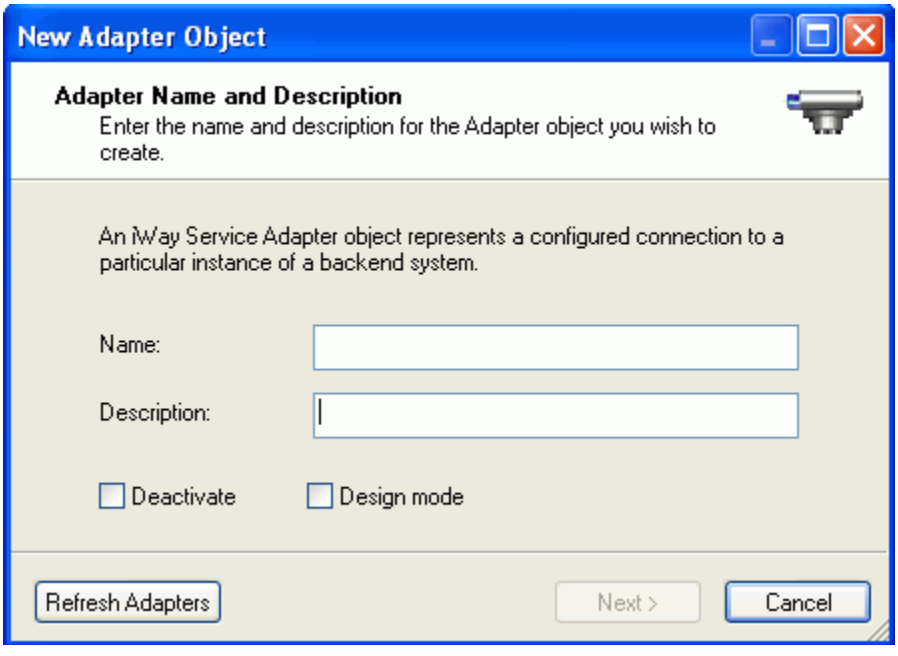
To use an adapter in a process flow, you must first configure a target for the adapter within iWay Explorer. For more information on configuring a target see *Connecting to Oracle E-Business Suite* on page 53.

Before you begin, ensure that you have created a project and created a process for that project. You can create a project by right-clicking the Processes folder in your project and selecting New Process from the context menu. For more information, see the *iWay Integration Tools (iIT) Designer User's Guide.*

Procedure: How to Add a Previously Defined Adapter

1. Select the process to which you want to add the adapter.
2. From the button bar, click and drag the Adapter object icon to the workspace.

The New Adapter Object dialog box appears, as shown in the following image.



a. Provide a descriptive name and a brief description (optional) for the Adapter object.

b. If you have updated adapter target information in iWay Explorer, click *Refresh Adapters*.

Note: The Deactivate option allows you to configure the object, but suppress its function within the process flow (usually used for debugging). The Design Mode option allows you to insert the object without configuring it, so that it acts as a placeholder in the process flow.

3. Click *Next*.

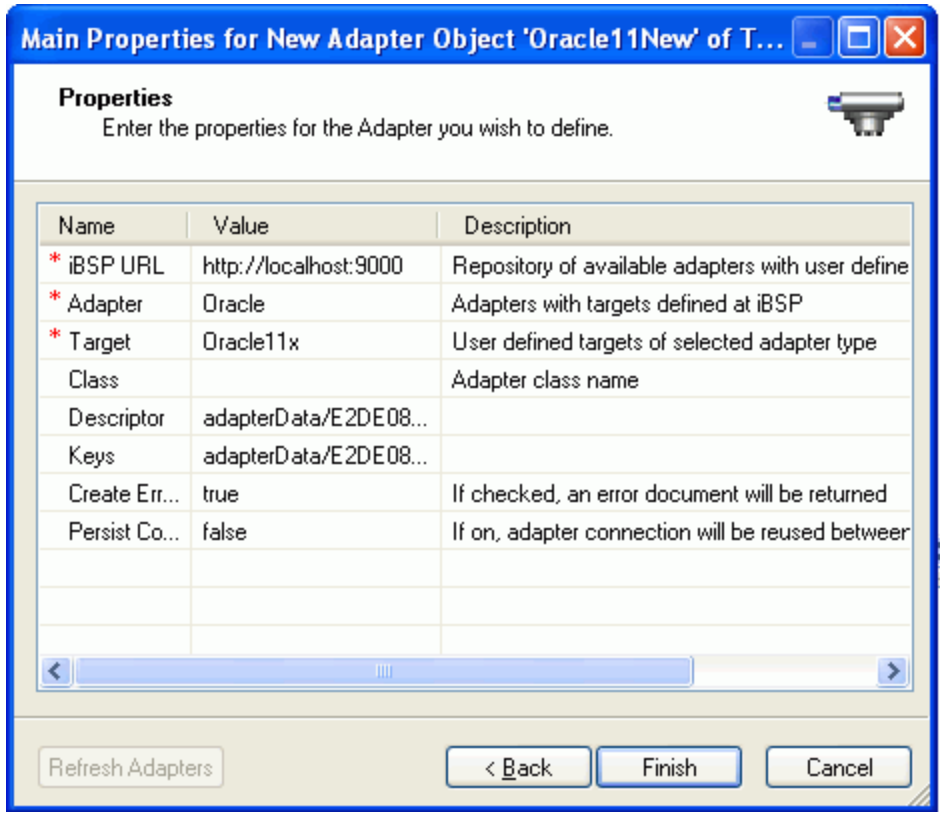
   You are prompted to select the adapter type from the Defined Adapters list.

4. Select the adapter target from the Defined Adapters drop-down list.

   If you do not see the adapter target in the list, you must go through the process of defining the adapter. For instructions, see *How to Define a New Adapter in iWay Integration Tools Designer* on page 200.

5. Click *Next*.

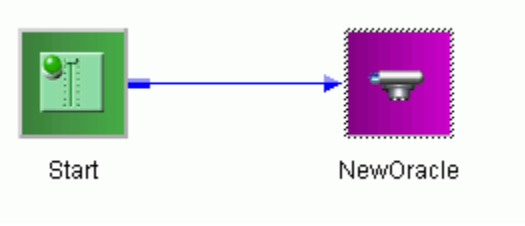
The Properties dialog box for the adapter you selected opens. The following image is an example of the adapter Properties dialog box.



In this dialog box, you can review and change the properties associated with the target. To change a property, click in the field and edit it directly. If multiple dialog boxes appear, continue to click *Next* to review all of the properties.

6. Click *Finish*.

An icon representing the Adapter object appears in the workspace and the adapter is available to your iWay process flow. For information on defining relationships between the Adapter object and other objects in your iWay process flow, see the *iWay Integration Tools (iIT) Designer User's Guide.*.

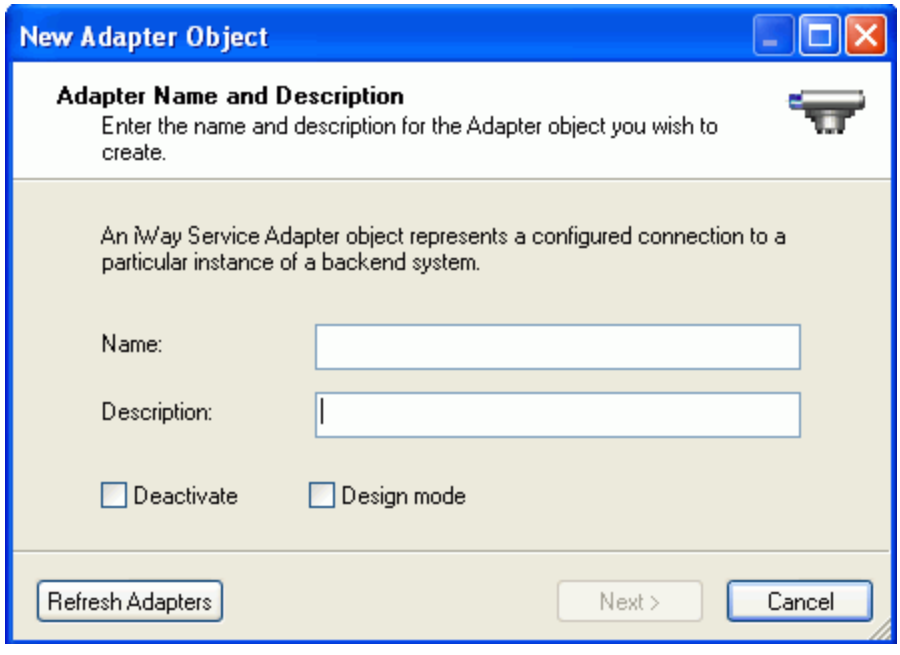


Procedure: How to Define a New Adapter in iWay Integration Tools Designer

You can create a new adapter object that has *not* been previously defined in the repository from the New Adapter Object dialog box.

1. Select the process to which you want to add the adapter object.
2. From the button bar, click and drag the Adapter Object icon to the workspace.

The New Adapter Object - Adapter Name and Description dialog box opens, as shown in the following image.



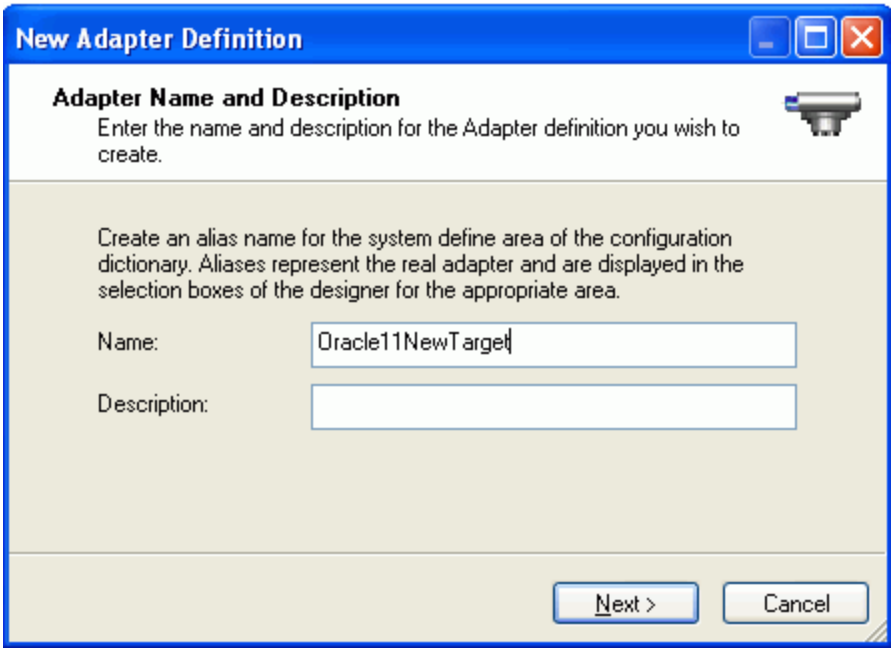
a. Provide a descriptive name and a brief description (optional) for the Adapter object.

b. If you have updated adapter target information in iWay Explorer, click *Refresh Adapters*.

Note: The Deactivate option allows you to configure the object, but suppress its function within the process flow (usually used for debugging). The Design Mode option allows you to insert the object without configuring it, so that it acts as a placeholder in the process flow.

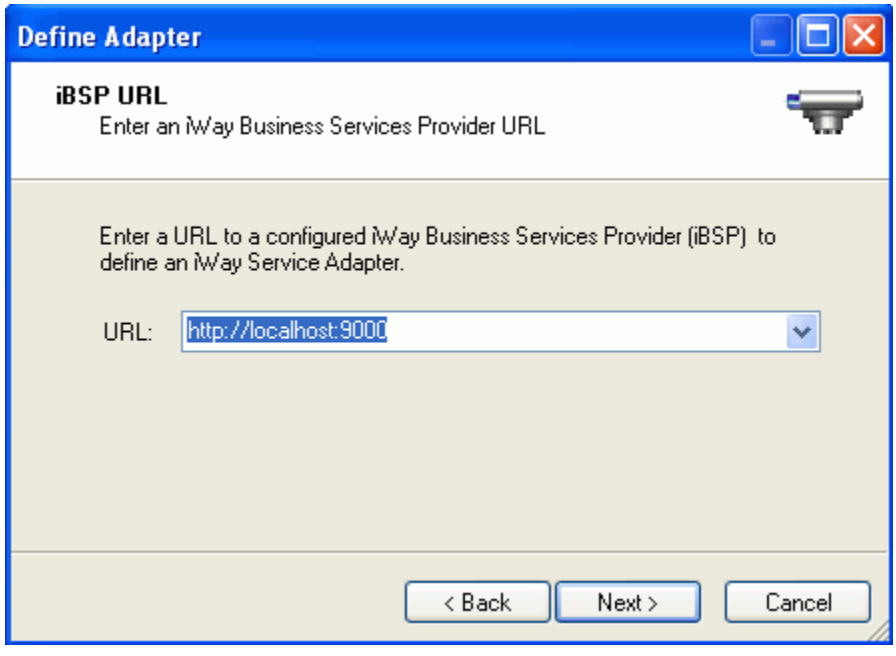
3. Click *Next*.
4. In the Adapter Type dialog box, click *New*.

The New Adapter Definition - Adapter Name and Description dialog box opens, as shown in the following image.



5. Type a unique, descriptive name to be used as an alias and a brief description (optional) for the adapter you are defining.
6. Click *Next*.

The Define Adapter iBSP URL dialog box opens, as shown in the following image.



7. Select an available iBSP URL from the drop-down list or type a URL in the following format:

   `http://host:port`

   where

   `host`

   Is the name of the machine on which iWay Service Manager is installed.

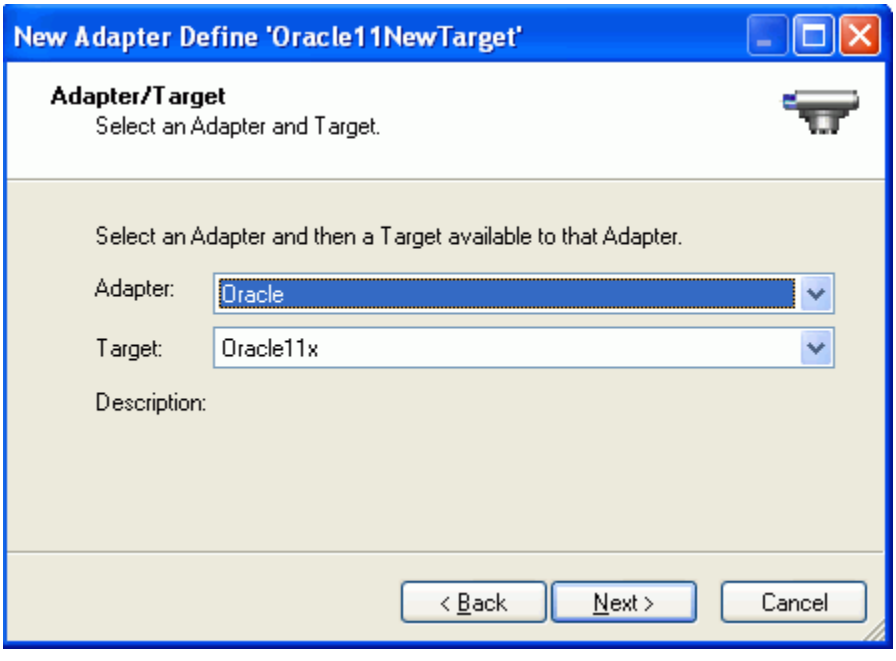
   `port`

   Is the port number on which iBSP listens for SOAP requests. The default is 9000.

   The iBSP URL allows you to access targets defined in the iBSP repository.

8. Click *Next*.

The Adapter/Target dialog box opens, as shown in the following image.

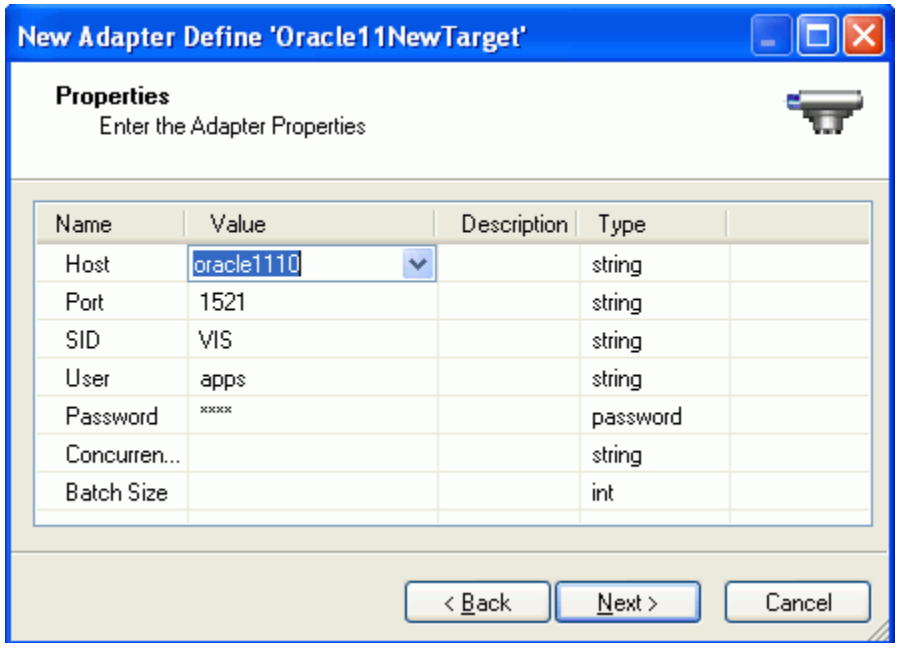


a. From the Adapter drop-down list, select the adapter.

b. From the Target drop-down list, select the target you want to use in your iWay process flow.

9. Click *Next*.

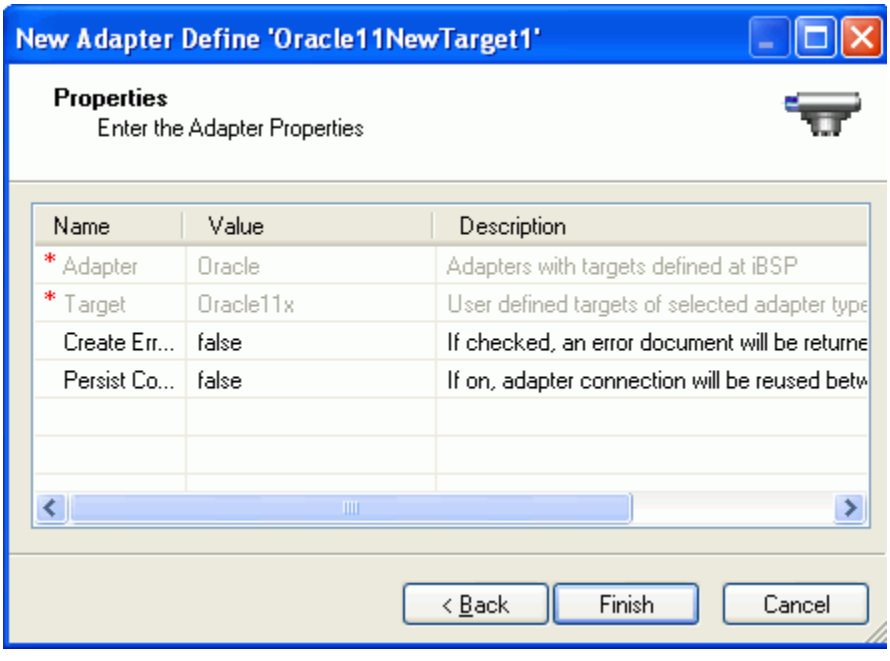
The descriptor properties for the adapter and target you selected appear. The following image shows an example of the Properties dialog box.



You can review and change the parameters associated with the target. To change a parameter, click in the field and edit it directly. If multiple dialog boxes appear, continue to click *Next* to review all of the parameters.

10. Click *Next* at the last New Adapter Define dialog box.

The adapter properties dialog box appears. You can choose to create an error document and to persist connections, as shown in the following image.

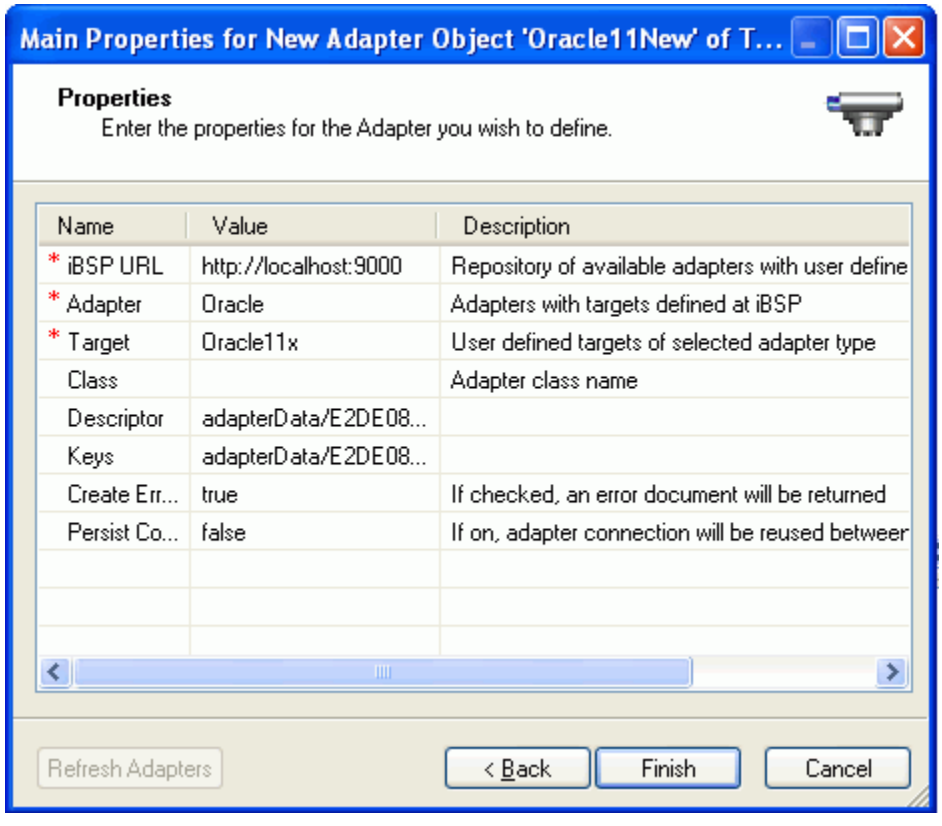


11. Click *Finish.*

    The Adapter Type dialog box appears with the newly created adapter in the Defined Adapters field.

12. Click *Next*.

The Main Properties dialog box appears. You can review and change the properties for the new adapter definition. For example, if you want to create an error document, change the value to *true*.

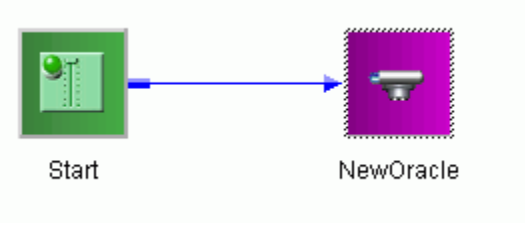


13. To use this adapter to create the Adapter object, click *Finish*.

    The adapter and target properties for the Adapter object appear in the Properties dialog box. You can review all parameters associated with the target again. Continue to click *Next* to move through the dialog boxes.

14. At the final Properties dialog box, click *Finish*.

An icon representing the Adapter object appears in the workspace, and the adapter is available to your iWay process flow. For information on defining relationships between the Adapter object and other objects in your iWay process flow, see the *iWay Integration Tools (iIT) Designer User's Guide.*.



Appendix C

Supported Interface Tables for Oracle Release Apps 11i

The Oracle Application environment consists of different functional components. This section describes the agents that constitute Oracle functional components.

In this appendix:

Application Adapter for Oracle E-Business Suite Supported Interface Tables

This section lists Oracle published interface tables available in release 11.5.5 of Oracle E-Business Suite that are supported by iWay Application Adapter for Oracle E-Business Suite.

Using Application Explorer, you can browse interface table metadata.

General Ledger

GL_INTERFACE

BUDGET_INTERFACE

GL_DAILY_RATES_INTERFACE

WIP Work Order

WIP_JOB_SCHEDULE_INTERFACE

WIP_JOB_DTLS_INTERFACE

WIP Move

WIP_MOVE_TXN_INTERFACE

CST_COMP_SNAP_INTERFACE

Payables

AP_INVOICES_INTERFACE

AP_INVOICE_LINE_INTERFACE

AP_EXPENSE_FEED_LINES

Receivables

RA_CUSTOMER_INTERFACE

RA_CUSTOMER_PROFILES_INTERFACE

RA_CONTACT_PHONES_INTERFACE

RA_CUSTOMER_BANKS_INTERFACE

RA_CUST_PAY_METHOD_INTERFACE

RA_INTERFACE_LINES

RA_INTERFACE_SALESCREDITS

RA_INTERFACE_DISTRIBUTIONS

AR_PAYMENTS_INTERFACE_ALL

Cash Management

CE_STATEMENT_HEADERS_INT_ALL

CE_STATEMENT_LINE_INTERFACE

Fixed Assets

FA_MASS_ADDITIONS

Manufacturing and Distribution

MTL_TRANSACTIONS_INTERFACE

MTL_SERIAL_NUMBERS_INTERFACE

MTL_TRANSACTION_LOTS_INTERFACE

MTL_DEMAND_INTERFACE

MTL_ITEM_QUANTITIES_VIEW

MTL_USER_SUPPLY

MTL_USER_DEMAND

MTL_REPLENISH_HEADERS_INT

MTL_SYSTEM_ITEMS_INTERFACE

MTL_ITEM_REVISIONS_INTERFACE

MTL_REPLENISH_LINES_INT

MTL_CI_INTERFACE

MTL_CI_XREFS_INTERFACE

Engineering and Bills of Material

BOM_BILL_OF_MTLS_INTERFACE

BOM_INVENTORY_COMPS_INTERFACE

BOM_REF_DESGS_INTERFACE

BOM_SUB_COMPS_INTERFACE

MTL_ITEM_REVISIONS_INTERFACE

BOM_OP_ROUTINGS_INTERFACE

BOM_OP_SEQUENCES_INTERFACE

BOM_OP_RESOURCES_INTERFACE

MTL_RTG_ITEM_REVS_INTERFACE

ENG_ENG_CHANGES_INTERFACE

ENG_ECO_REVISIONS_INTERFACE

ENG_REVISED_ITEMS_INTERFACE

BOM_INVENTORY_COMPS_INTERFACE

BOM_REF_DESGS_INTERFACE

BOM_SUB_COMPS_INTERFACE

Cost Management

CST_PC_ITEM_COST_INTERFACE

CST_PC_COST_DET_INTERFACE

Master Scheduling and Oracle Supply Chain

MRP_FORECAST_INTERFACE

MRP_SCHEDULE_INTERFACE

MRP_RELIEF_INTERFACE

WIP_JOB_SCHEDULE_INTERFACE

PO_REQUISITIONS_INTERFACE

PO_RESCHEDULE_INTERFACE

SO_HEADERS_INTERFACE_ALL

SO_HEADER_ATTRIBUTES_INTERFACE

SO_LINES_INTERFACE_ALL

SO_LINE_ATTRIBUTES_INTERFACE

SO_LINE_DETAILS_INTERFACE

SO_SALES_CREDITS_INTERFACE

SO_PRICE_ADJUSTMENTS_INTERFACE

WSH_DELIVERIES_INTERFACE

WSH_PACKED_CONTAINER_INTERFACE

WSH_PICKING_DETAILS_INTERFACE

WSH_FREIGHT_CHARGES_INTERFACE

Purchasing

PO_REQUISITIONS_INTERFACE

PO_REQ_DIST_INTERFACE

PO_RESCHEDULE_INTERFACE

PO_HEADERS_INTERFACE

PO_LINES_INTERFACE

Quality

RCV_HEADERS_INTERFACE

RCV_TRANSACTIONS_INTERFACE

QA_RESULTS_INTERFACE

Human Resources

GHR_INTERFACE

PAY_GB_TAX_CODE_INTERFACE

PAY_GL_INTERFACE

PAY_IE_TAX_BODY_INTERFACE

PAY_IE_TAX_HEADER_INTERFACE

PAY_IE_TAX_TRAILER_INTERFACE

PSP_DISTRIBUTION_INTERFACE

Customer Relationship Management

JTF_TTY_GEO_WEBADI_INTERFACE

CN_SCA_HEADERS_INTERFACE_ALL

CN_SCA_LINES_INTERFACE_ALL

Appendix D

Sample WSDL File

This section shows an example of a WSDL file generated from a web service using iWay Explorer.

In this appendix:

Sample WSDL File

The following is the full text of a sample WSDL file.

Example: **WSDL File Generated From A Web Service**

```
<definitions xmlns:tns="urn:schemas-iwaysoftware-com:iwse"
targetNamespace="urn:schemas-iwaysoftware-com:iwse"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:m11="urn:iwaysoftware:ibse:jul2003:test:response"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:m1="urn:iwaysoftware:ibse:jul2003:test"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"><types><xs:schema
targetNamespace="urn:schemas-iwaysoftware-com:iwse"
elementFormDefault="qualified"><xs:element
name="ibsinfo"><xs:complexType><xs:sequence><xs:element type="xs:string"
name="service"/><xs:element type="xs:string" name="method"/><xs:element
type="xs:string" name="license"/><xs:element type="xs:string"
minOccurs="0" name="disposition"/><xs:element type="xs:string"
minOccurs="0" name="Username"/><xs:element type="xs:string" minOccurs="0"
name="Password"/><xs:element type="xs:string" minOccurs="0"
name="language"/></xs:sequence></xs:complexType>
        </xs:element><xs:element
```

```
name="adapterexception"><xs:complexType><xs:sequence><xs:element
type="xs:string"
name="error"/></xs:sequence></xs:complexType></xs:element>
      </xs:schema><xs:schema xml:lang="en"
targetNamespace="urn:iwaysoftware:ibse:jul2003:test"
attributeFormDefault="unqualified"
xmlns:m1="urn:iwaysoftware:ibse:jul2003:test"
elementFormDefault="qualified"><xs:element
name="test"><xs:complexType><xs:sequence><xs:element
name="ORACLE"><xs:complexType><xs:sequence><xs:element
name="WIP_JOB_SCHEDULE_INTERFACE"><xs:complexType><xs:sequence><xs:elemen
t minOccurs="1" name="LAST_UPDATE_DATE" maxOccurs="1"/><xs:element
minOccurs="0" name="LAST_UPDATED_BY" maxOccurs="1"/><xs:element
minOccurs="1" name="CREATION_DATE" maxOccurs="1"/><xs:element
minOccurs="0" name="CREATED_BY" maxOccurs="1"/><xs:element minOccurs="0"
name="LAST_UPDATE_LOGIN" maxOccurs="1"/><xs:element minOccurs="0"
name="REQUEST_ID" maxOccurs="1"/><xs:element minOccurs="0"
name="PROGRAM_ID" maxOccurs="1"/><xs:element minOccurs="0"
name="PROGRAM_APPLICATION_ID" maxOccurs="1"/><xs:element minOccurs="0"
name="PROGRAM_UPDATE_DATE" maxOccurs="1"/><xs:element minOccurs="0"
name="GROUP_ID" maxOccurs="1"/><xs:element minOccurs="0"
name="SOURCE_CODE" maxOccurs="1"/><xs:element minOccurs="0"
name="SOURCE_LINE_ID" maxOccurs="1"/><xs:element minOccurs="0"
name="PROCESS_TYPE" maxOccurs="1"/><xs:element minOccurs="0"
name="ORGANIZATION_ID" maxOccurs="1"/><xs:element minOccurs="1"
name="LOAD_TYPE" maxOccurs="1"/><xs:element minOccurs="0"
name="STATUS_TYPE" maxOccurs="1"/><xs:element minOccurs="0"
name="OLD_STATUS_TYPE" maxOccurs="1"/><xs:element minOccurs="0"
name="LAST_UNIT_COMPLETION_DATE" maxOccurs="1"/><xs:element minOccurs="0"
name="OLD_COMPLETION_DATE" maxOccurs="1"/><xs:element minOccurs="0"
name="PROCESSING_WORK_DAYS" maxOccurs="1"/><xs:element minOccurs="0"
name="DAILY_PRODUCTION_RATE" maxOccurs="1"/><xs:element minOccurs="0"
```

```
name="LINE_ID" maxOccurs="1"/><xs:element minOccurs="0"
name="PRIMARY_ITEM_ID" maxOccurs="1"/><xs:element minOccurs="0"
name="BOM_REFERENCE_ID" maxOccurs="1"/><xs:element minOccurs="0"
name="ROUTING_REFERENCE_ID" maxOccurs="1"/><xs:element minOccurs="0"
name="BOM_REVISION_DATE" maxOccurs="1"/><xs:element minOccurs="0"
name="ROUTING_REVISION_DATE" maxOccurs="1"/><xs:element minOccurs="0"
name="WIP_SUPPLY_TYPE" maxOccurs="1"/><xs:element minOccurs="0"
name="CLASS_CODE" maxOccurs="1"/><xs:element minOccurs="0"
name="LOT_NUMBER" maxOccurs="1"/><xs:element minOccurs="0"
name="LOT_CONTROL_CODE" maxOccurs="1"/><xs:element minOccurs="0"
name="JOB_NAME" maxOccurs="1"/><xs:element minOccurs="0"
name="DESCRIPTION" maxOccurs="1"/><xs:element minOccurs="0"
name="FIRM_PLANNED_FLAG" maxOccurs="1"/><xs:element minOccurs="0"
name="ALTERNATE_ROUTING_DESIGNATOR" maxOccurs="1"/><xs:element
minOccurs="0" name="ALTERNATE_BOM_DESIGNATOR" maxOccurs="1"/><xs:element
minOccurs="0" name="DEMAND_CLASS" maxOccurs="1"/><xs:element
minOccurs="0" name="START_QUANTITY" maxOccurs="1"/><xs:element
minOccurs="0" name="OLD_START_QUANTITY" maxOccurs="1"/><xs:element
minOccurs="0" name="WIP_ENTITY_ID" maxOccurs="1"/><xs:element
minOccurs="0" name="REPETITIVE_SCHEDULE_ID" maxOccurs="1"/><xs:element
minOccurs="0" name="ERROR" maxOccurs="1"/><xs:element minOccurs="0"
name="PARENT_GROUP_ID" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE_CATEGORY" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE1" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE2" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE3" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE4" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE5" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE6" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE7" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE8" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE9" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE10" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE11" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE12" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE13" maxOccurs="1"/><xs:element minOccurs="0"
```

```
name="ATTRIBUTE14" maxOccurs="1"/><xs:element minOccurs="0"
name="ATTRIBUTE15" maxOccurs="1"/><xs:element minOccurs="0"
name="INTERFACE_ID" maxOccurs="1"/><xs:element minOccurs="0"
name="LAST_UPDATED_BY_NAME" maxOccurs="1"/><xs:element minOccurs="0"
name="CREATED_BY_NAME" maxOccurs="1"/><xs:element minOccurs="0"
name="PROCESS_PHASE" maxOccurs="1"/><xs:element minOccurs="0"
name="PROCESS_STATUS" maxOccurs="1"/><xs:element minOccurs="0"
name="ORGANIZATION_CODE" maxOccurs="1"/><xs:element minOccurs="0"
name="FIRST_UNIT_START_DATE" maxOccurs="1"/><xs:element minOccurs="0"
name="FIRST_UNIT_COMPLETION_DATE" maxOccurs="1"/><xs:element
minOccurs="0" name="LAST_UNIT_START_DATE" maxOccurs="1"/><xs:element
minOccurs="0" name="SCHEDULING_METHOD" maxOccurs="1"/><xs:element
minOccurs="0" name="LINE_CODE" maxOccurs="1"/><xs:element minOccurs="0"
name="PRIMARY_ITEM_SEGMENTS" maxOccurs="1"/><xs:element minOccurs="0"
name="BOM_REFERENCE_SEGMENTS" maxOccurs="1"/><xs:element minOccurs="0"
name="ROUTING_REFERENCE_SEGMENTS" maxOccurs="1"/><xs:element
minOccurs="0" name="ROUTING_REVISION" maxOccurs="1"/><xs:element
minOccurs="0" name="BOM_REVISION" maxOccurs="1"/><xs:element
minOccurs="0" name="COMPLETION_SUBINVENTORY" maxOccurs="1"/><xs:element
minOccurs="0" name="COMPLETION_LOCATOR_ID" maxOccurs="1"/><xs:element
minOccurs="0" name="COMPLETION_LOCATOR_SEGMENTS"
maxOccurs="1"/><xs:element minOccurs="0" name="SCHEDULE_GROUP_ID"
maxOccurs="1"/><xs:element minOccurs="0" name="SCHEDULE_GROUP_NAME"
maxOccurs="1"/><xs:element minOccurs="0" name="BUILD_SEQUENCE"
maxOccurs="1"/><xs:element minOccurs="0" name="PROJECT_ID"
maxOccurs="1"/><xs:element minOccurs="0" name="PROJECT_NAME"
maxOccurs="1"/><xs:element minOccurs="0" name="TASK_ID"
maxOccurs="1"/><xs:element minOccurs="0" name="TASK_NAME"
maxOccurs="1"/><xs:element minOccurs="0" name="NET_QUANTITY"
maxOccurs="1"/><xs:element minOccurs="0" name="DESCRIPTIVE_FLEX_SEGMENTS"
maxOccurs="1"/><xs:element minOccurs="0" name="PROJECT_NUMBER"
maxOccurs="1"/><xs:element minOccurs="0" name="TASK_NUMBER"
maxOccurs="1"/><xs:element minOccurs="0" name="PROJECT_COSTED"
maxOccurs="1"/><xs:element minOccurs="0" name="END_ITEM_UNIT_NUMBER"
maxOccurs="1"/><xs:element minOccurs="0"
name="OVERCOMPLETION_TOLERANCE_TYPE" maxOccurs="1"/><xs:element
minOccurs="0" name="OVERCOMPLETION_TOLERANCE_VALUE"
maxOccurs="1"/><xs:element minOccurs="0" name="KANBAN_CARD_ID"
```

```
maxOccurs="1"/><xs:element minOccurs="0" name="PRIORITY"
maxOccurs="1"/><xs:element minOccurs="0" name="DUE_DATE"
maxOccurs="1"/><xs:element minOccurs="0" name="ALLOW_EXPLOSION"
maxOccurs="1"/><xs:element minOccurs="0" name="HEADER_ID"
maxOccurs="1"/><xs:element minOccurs="0" name="DELIVERY_ID"
maxOccurs="1"/><xs:element minOccurs="0" name="COPRODUCTS_SUPPLY"
maxOccurs="1"/><xs:element minOccurs="0" name="DUE_DATE_PENALTY"
maxOccurs="1"/><xs:element minOccurs="0" name="DUE_DATE_TOLERANCE"
maxOccurs="1"/><xs:element minOccurs="0" name="XML_DOCUMENT_ID"
maxOccurs="1"/><xs:element minOccurs="0" name="PARENT_WIP_ENTITY_ID"
maxOccurs="1"/><xs:element minOccurs="0" name="PARENT_JOB_NAME"
maxOccurs="1"/><xs:element minOccurs="0" name="ASSET_NUMBER"
maxOccurs="1"/><xs:element minOccurs="0" name="ASSET_GROUP_ID"
maxOccurs="1"/><xs:element minOccurs="0" name="ASSET_GROUP_SEGMENTS"
maxOccurs="1"/><xs:element minOccurs="0" name="PM_SCHEDULE_ID"
maxOccurs="1"/><xs:element minOccurs="0" name="REBUILD_ITEM_ID"
maxOccurs="1"/><xs:element minOccurs="0" name="REBUILD_ITEM_SEGMENTS"
maxOccurs="1"/><xs:element minOccurs="0" name="REBUILD_SERIAL_NUMBER"
maxOccurs="1"/><xs:element minOccurs="0" name="MANUAL_REBUILD_FLAG"
maxOccurs="1"/><xs:element minOccurs="0" name="SHUTDOWN_TYPE"
maxOccurs="1"/><xs:element minOccurs="0" name="NOTIFICATION_REQUIRED"
maxOccurs="1"/><xs:element minOccurs="0" name="WORK_ORDER_TYPE"
maxOccurs="1"/><xs:element minOccurs="0" name="OWNING_DEPARTMENT"
maxOccurs="1"/><xs:element minOccurs="0" name="OWNING_DEPARTMENT_CODE"
maxOccurs="1"/><xs:element minOccurs="0" name="ACTIVITY_TYPE"
maxOccurs="1"/><xs:element minOccurs="0" name="ACTIVITY_CAUSE"
maxOccurs="1"/><xs:element minOccurs="0" name="TAGOUT_REQUIRED"
maxOccurs="1"/><xs:element minOccurs="0" name="PLAN_MAINTENANCE"
maxOccurs="1"/><xs:element minOccurs="0" name="DATE_RELEASED"
maxOccurs="1"/><xs:element minOccurs="0" name="REQUESTED_START_DATE"
maxOccurs="1"/></xs:sequence></xs:complexType></xs:element></xs:sequence>
</xs:complexType></xs:element></xs:sequence></xs:complexType></xs:element
>


      </xs:schema><xs:schema xml:lang="en"
targetNamespace="urn:iwaysoftware:ibse:jul2003:test:response"
attributeFormDefault="unqualified"
xmlns:m11="urn:iwaysoftware:ibse:jul2003:test:response"
elementFormDefault="qualified"><xs:element
name="testResponse"><xs:complexType><xs:sequence><xs:element
name="ORACLE"><xs:complexType><xs:sequence><xs:element minOccurs="0"
name="Processed_table" maxOccurs="unbounded"/><xs:element minOccurs="0"
name="submit_response_id" maxOccurs="unbounded"/><xs:element
name="Return_Code"/></xs:sequence></xs:complexType></xs:element>


              </xs:sequence><xs:attribute type="xs:string"
use="required" name="cid"/></xs:complexType></xs:element></xs:schema>


   </types><message name="testIn"><part element="m1:test"
name="parameters"/>


   </message><message name="testOut"><part element="m11:testResponse"
name="parameters"/>
```

```
   </message><message name="testserviceHeader"><part
element="tns:ibsinfo" name="header"/>

   </message><message name="AdapterException"><part
element="tns:adapterexception" name="fault"/>

   </message><portType name="testserviceSoap"><operation
name="test"><documentation/><input message="tns:testIn"/><output
message="tns:testOut"/><fault message="tns:AdapterException"
name="AdapterExceptionFault"/></operation>

   </portType><binding type="tns:testserviceSoap"
name="testserviceSoap"><soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/><operation
name="test"><soap:operation style="document"
soapAction="testservice.testRequest@test@@"/><input><soap:body
use="literal"/><soap:header part="header" message="tns:testserviceHeader"
use="literal"/>

         </input><output><soap:body use="literal"/>

         </output><fault name="AdapterExceptionFault"><soap:fault
use="literal" name="AdapterExceptionFault"/></fault></operation>

   </binding><service
name="testservice"><documentation>testservice</documentation><port
binding="tns:testserviceSoap" name="testserviceSoap1"><soap:address
location="http://111KLEINMAN.ibi.com:7001/ibse/IBSEServlet/XDSOAPRouter"/
></port></service></definitions>
```

Feedback

Customer success is our top priority. Connect with us today!

Information Builders Technical Content Management team is comprised of many talented individuals who work together to design and deliver quality technical documentation products. Your feedback supports our ongoing efforts!

You can also preview new innovations to get an early look at new content products and services. Your participation helps us create great experiences for every customer.

To send us feedback or make a connection, contact Sarah Buccellato, Technical Editor, Technical Content Management at *Sarah_Buccellato@ibi.com.*

To request permission to repurpose copyrighted material, please contact Frances Gambino, Vice President, Technical Content Management at *Frances_Gambino@ibi.com.*

Information Builders

iWay

iWay Application Adapter for Oracle E-Business Suite User's Guide

Version 7.0.x and Higher

DN3502259.0418

Information Builders, Inc.
Two Penn Plaza
New York, NY 10121-2898