



webMethods Oracle Applications Adapter

User's Guide

VERSION 6.0

webMethods Administrator, webMethods Broker, webMethods Dashboard, webMethods Developer, webMethods Fabric, webMethods Glue, webMethods Installer, webMethods Integration Server, webMethods Mainframe, webMethods Manager, webMethods Mobile, webMethods Modeler, webMethods Monitor, webMethods Optimize, webMethods Portal, webMethods Trading Networks, webMethods Workflow, and the webMethods logo are trademarks of webMethods, Inc. "webMethods" is a registered trademark of webMethods, Inc.

Acrobat, Adobe, and Reader are registered trademarks of Adobe Systems Incorporated. Amdocs and ClarifyCRM are registered trademarks of Amdocs Ltd. Ariba is a registered trademark of Ariba Inc. BEA is a registered trademark, and BEA WebLogic Platform and BEA WebLogic Server are trademarks of BEA Systems, Inc. BMC Software and PATROL are registered trademarks of BMC Software, Inc. BroadVision is a registered trademark of BroadVision, Inc. Chem eStandards and CIDX are trademarks of Chemical Industry Data Exchange. Unicenter is a registered trademark of Computer Associates International, Inc. PopChart is a registered trademark of CORDA Technologies, Inc. Kenan and Arbor are registered trademarks of CSG Systems, Incorporated. SNAP-IX is a registered trademark, and Data Connection is a trademark of Data Connection Ltd. DataDirect, DataDirect Connect, and SequeLink are registered trademarks of DataDirect Technologies. D&B and D-U-N-S are registered trademarks of D&B, Inc. Entrust is a registered trademark of Entrust. Hewlett-Packard, HP, HP-UX, and OpenView are trademarks of Hewlett-Packard Company. i2 is a registered trademark of i2 Technologies, Inc. AIX, AS/400, CICS, DB2, IBM, Infoprint, Informix, MQSeries, OS/390, OS/400, RACF, RS/6000, SQL/400, S/390, System/390, VTAM, and WebSphere are registered trademarks; and Communications System for Windows NT, IMS, MVS, SQL/DS, Universal Database, and z/OS are trademarks of IBM Corporation. InnoDB is a trademark of Innobase Oy. JBoss and JBoss Group are trademarks of Marc Fleury under operation by JBoss Group, LLC. J.D. Edwards and OneWorld are registered trademarks, and WorldSoftware is a trademark of J.D. Edwards. Linux is a registered trademark of Linus Torvalds and others. X Window System is a trademark of Massachusetts Institute of Technology. MetaSolv is a registered trademark of Metasolv Software, Inc. ActiveX, Microsoft, Outlook, Visual Basic, Windows, and Windows NT are registered trademarks; and SQL Server is a trademark of Microsoft Corporation. MySQL is a registered trademark of MySQL AB. Teradata is a registered trademark of NCR. Netscape is a registered trademark of Netscape Communications Corporation. New Atlanta and ServletExec are trademarks of New Atlanta Communications, LLC. CORBA is a registered trademark of Object Management Group, Inc. UNIX is a registered trademark of Open Group. Oracle is a registered trademark of Oracle Corporation. PeopleSoft and Vantive are registered trademarks, and PeopleSoft Pure Internet Architecture is a trademark of PeopleSoft, Inc. Intranet and Portal are trademarks of Portal Software, Inc. RosettaNet is a trademark of "RosettaNet," a non-profit organization. SAP and R/3 are trademarks or registered trademarks of SAP AG. Siebel is a trademark of Siebel Systems, Inc. SPARC and SPARCStation are trademarks of SPARC International, Inc. SSA Global is a trademark and SSA Baan is a registered trademark of SSA Global Technologies, Inc. EJB, Enterprise JavaBeans, Java, Java Naming and Directory Interface, JavaServer Pages, JDBC, JSP, J2EE, Solaris, Sun Microsystems, and SunSoft are trademarks of Sun Microsystems, Inc. SWIFT and SWIFTNet are trademarks of S.W.I.F.T. SCRL. Sybase is a registered trademark of Sybase, Inc. UCCnet is a trademark of UCCnet. eBusinessReady is a trademark of Uniform Code Council, Inc. (UCC) and Drummond Group, Inc. (DGI). Verisign is a registered trademark of Verisign. VERITAS, VERITAS SOFTWARE, and VERITAS Cluster Server are trademarks of VERITAS Software. W3C is a registered trademark of World Wide Web Consortium.

All other marks are the property of their respective owners.

Copyright © 2004 by webMethods, Inc. All rights reserved, including the right of reproduction in whole or in part in any form.

Document ID: ADAPTER-ORACLEAPPS-UG-60-20040716

Contents

About This Guide	7
Document Conventions	7
Additional Information	8
Chapter 1. About the Oracle Applications Adapter	9
Overview of the Adapter	10
Architecture and Components	10
Package Management	13
Adapter Connections	14
Using the JDBC Driver to Connect to Oracle Applications	14
Transaction Management of Oracle Applications Adapter Connections	15
Connection Pools	15
Run-Time Behavior of Connection Pools	15
Built-In Services For Connections	16
Changing the Connection Associated With an Adapter Service or Notification at Design Time ..	16
Transaction Definitions	17
Adapter Services	18
Adapter Service Templates	18
Predefined Transaction Services	20
Adapter Notifications	20
webMethods Manager Support for the Oracle Applications Adapter	21
Viewing Different Perspectives of the webMethods Developer	22
Chapter 2. Package Management	23
Overview	24
Oracle Applications Adapter Package Management	24
Package Dependency Requirements and Guidelines	25
Enabling and Disabling Packages	26
Loading, Reloading, and Unloading Packages	26
Reloading Packages Manually	26
Unloading Packages	26
Setting Package Dependencies	27
Group Access Control	27
Oracle Applications Adapter in a Clustered Environment	28
What is webMethods Integration Server Clustering?	28
Configuring the Oracle Applications Adapter	28
Replicating Packages to webMethods Integration Servers	28

Disabling the Redirection of Administrative Services	29
Clustering Considerations and Requirements	29
Requirements for Each Integration Server in a Cluster	30
Considerations When Installing Oracle Applications Adapter Packages	31
Considerations When Configuring Connections with Connection Pooling Enabled	31
Scheduling Service Invocations in a Clustered Environment	31
Chapter 3. Adapter Connections	33
Overview	34
Before Configuring or Managing Adapter Connections	34
Setting Up Your Adapter's Environment	35
Configuring Adapter Connections	35
Viewing Adapter Connection Parameters	40
Editing Adapter Connections	40
Copying Adapter Connections	41
Deleting Adapter Connections	42
Enabling Adapter Connections	43
Disabling Adapter Connections	43
Chapter 4. Transaction Definitions	45
Overview	46
Before Working with Transaction Definitions	46
Importing Transaction Definitions	46
Exporting Transaction Definitions	48
Creating IS-to-Oracle Applications Transaction Definitions	50
Creating Oracle Applications-to-IS Transaction Definitions	51
Editing IS-to-Oracle Applications Transaction Definitions	52
Editing Oracle Applications-to-IS Transaction Definitions	53
Deleting Transaction Definitions	54
Working with SQL Trees in Oracle Applications-to-IS Transactions	55
SQL SELECT Statement Syntax	55
Stored Procedure Syntax	56
SQL Tree Example	56
Parent Level Alias	57
Child Alias 1	58
Child Alias 2	58
Results	59

Chapter 5. Adapter Services	61
Overview	62
Before Configuring or Managing Adapter Services	62
Using Oracle Applications Adapter Services	63
Configuring Insert SQL Services	64
Configuring Compound Select SQL Services	65
Configuring Simple SQL Services	66
Testing Adapter Services	68
Viewing Adapter Services	69
Editing Adapter Services	69
Deleting Adapter Services	70
Enabling Automatic Data Validation	70
Reloading Adapter Values in Developer	71
Chapter 6. Adapter Notifications	73
Overview	74
Before Configuring or Managing Notifications	74
Configuring Notifications	75
Managing Notifications	76
Exporting Configured Adapter Notifications	78
Viewing Notifications	79
Editing Notifications	79
Deleting Notifications	80
Validating Adapter Notification Values	80
Reloading Adapter Values	81
Chapter 7. Logging and Exception Handling	83
Overview	84
Oracle Applications Adapter Message Logging	84
Handling Oracle Applications Errors	85
Oracle Applications Adapter Error Codes	86
Appendix A. Built-In Transaction Management Services	97
Transaction Management Overview	98
Transactions	98
Implicit and Explicit Transactions	98
Implicit Transactions	98
Explicit Transactions	99
Changing the Integration Server's Transaction Timeout Interval	100
Index	101

About This Guide

This guide describes how to configure and use the webMethods Oracle Applications Adapter Version 6.0. It contains information for webMethods administrators and application developers who want to exchange data with Oracle Applications systems.

To use this guide effectively, you should be familiar with:

- The Oracle Applications system and how your organization uses specific modules and business transactions
- Database concepts and SQL
- The terminology and basic operations of your operating system
- webMethods Integration Server and Developer basic concepts and tasks

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
<i>Italic</i>	Identifies variable information that you must supply or change based on your specific situation or environment. Identifies terms the first time they are defined in text. Also identifies service input and output variables.
Narrow font	Identifies storage locations for services on the webMethods Integration Server using the convention <code>folder.subfolder:service</code> .
Typewriter font	Identifies characters and values that you must type exactly or messages that the system displays on the console.
UPPERCASE	Identifies keyboard keys. Keys that you must press simultaneously are joined with the “+” symbol.
\	Directory paths use the “\” directory delimiter unless the subject is UNIX-specific.
[]	Optional keywords or values are enclosed in []. Do not type the [] symbols in your own code.

Additional Information

The webMethods Advantage Web site at <http://advantage.webmethods.com> provides you with important sources of information about the webMethods Integration Server:

- **Troubleshooting Information.** webMethods provides troubleshooting information for many webMethods components in the [webMethods Knowledge Base](#).
- **Documentation Feedback.** To provide documentation feedback to webMethods, go to the [Documentation Feedback Form](#) on the [webMethods Bookshelf](#).
- **Additional Documentation.** All webMethods documentation is available on the [webMethods Bookshelf](#).

About the Oracle Applications Adapter

■ Overview of the Adapter	10
■ Architecture and Components	10
■ Package Management	13
■ Adapter Connections	14
■ Transaction Definitions	17
■ Adapter Services	18
■ Adapter Notifications	20
■ webMethods Manager Support for the Oracle Applications Adapter	21
■ Viewing Different Perspectives of the webMethods Developer	22

Overview of the Adapter

The webMethods Oracle Applications Adapter is an add-on to the webMethods Integration Platform that allows you to exchange data with Oracle Applications systems. The adapter provides seamless and real-time communications to and from the Oracle Applications system without requiring changes to the existing security infrastructure.

Using the Oracle Applications Adapter, webMethods Integration Server clients can run services that execute transactions to retrieve data from, and insert and update data into, Oracle Applications systems. For example, you can use the Oracle Applications Adapter in a flow that receives an XML-based purchase order and then inserts a sales order into an Oracle Applications system.

The adapter also provides user interfaces that enable you to configure and manage adapter connections, adapter services, adapter notifications, and transaction definitions.

Sample transaction definitions are available that you can use when configuring your own adapter services. You can use the samples as they are provided, or you can modify them or create new transaction definitions customized for your needs. This flexibility facilitates integration with highly customized Oracle Applications.

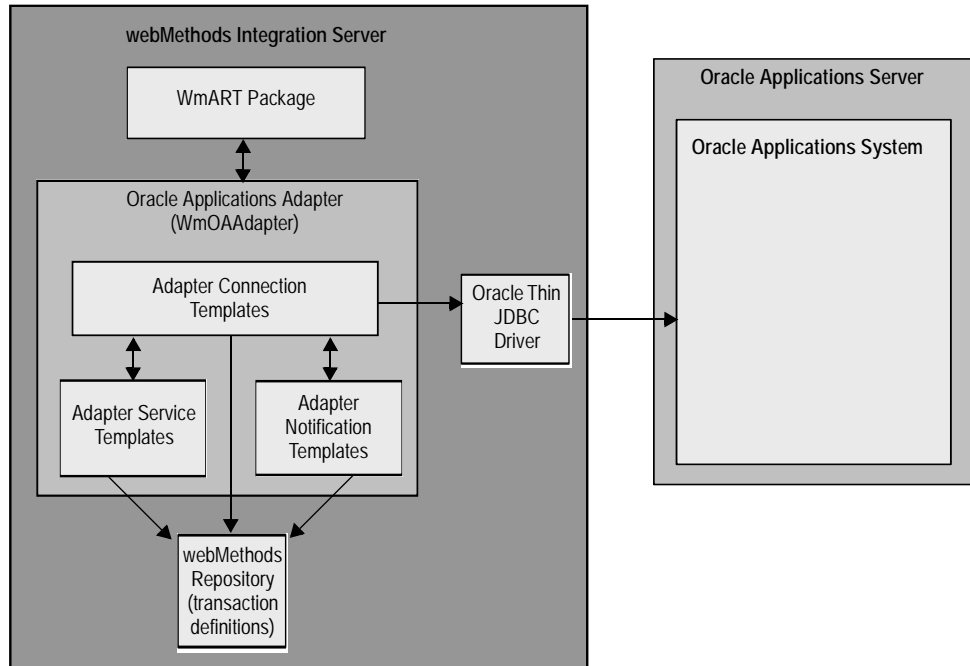
For Oracle Applications integrations that are not highly customized, there are sets of predefined transaction services that are available as add-ons to the webMethods Oracle Applications Adapter. Predefined transaction services are complete, ready-to-use flow services that simplify the process of integrating with Oracle Applications systems. In most cases, you can use these services as they are supplied. You only need to modify a connection to specify your Oracle Applications system connection information.

Architecture and Components

The Oracle Applications Adapter provides a set of user interfaces, services, and templates that enable you to create integrations with Oracle Applications systems. The adapter is provided as a single package that must be installed on the webMethods Integration Server. For detailed installation instructions and software requirements, see the *webMethods Oracle Applications Adapter Installation Guide*.

Because the Oracle Applications Adapter uses an Oracle Thin JDBC driver to access the data in an Oracle Applications system, the adapter requires the driver to be installed and loaded in the packages directory of the Integration Server.

The following diagram shows at a high level how the adapter components connect to an Oracle Applications system.



- **webMethods Integration Server.** The Oracle Applications Adapter is installed and runs on the Integration Server.
- **WmART Package.** The WmART package provides a common framework for version 6.0 and later adapters to use the Integration Server’s functionality, making the Integration Server the run-time environment for the Oracle Applications Adapter. The WmART package is installed with the Integration Server.
- **Oracle Applications Adapter.** The Oracle Applications Adapter is delivered as a single package called WmOAAAdapter.

The Oracle Applications Adapter provides Integration Server Administrator and webMethods Developer user interfaces that enable you to configure and manage adapter connections, adapter services, adapter notifications, and transaction definitions.

- **Adapter Connection Templates.** Adapter connections enable the Oracle Applications Adapter to connect to Oracle Applications systems. You must configure an adapter connection before you can configure adapter services or notifications.

The adapter provides a template for adapter connections in the Integration Server Administrator. For a detailed description of adapter connections, see [“Adapter Connections” on page 14](#).

- **Adapter Service Templates.** Adapter services enable the Integration Server to initiate and perform operations on the Oracle Applications system.

The adapter provides templates for adapter services in the webMethods Developer. For a detailed description of adapter services, see [“Adapter Services” on page 18](#).

- **Adapter Notification Templates.** Adapter notifications enable the Oracle Applications Adapter to process data from an event that has occurred on an Oracle Applications system. The Oracle Applications Adapter uses a polling mechanism that supports polling services to provide Oracle Applications-to-webMethods Integration Server notification capabilities.

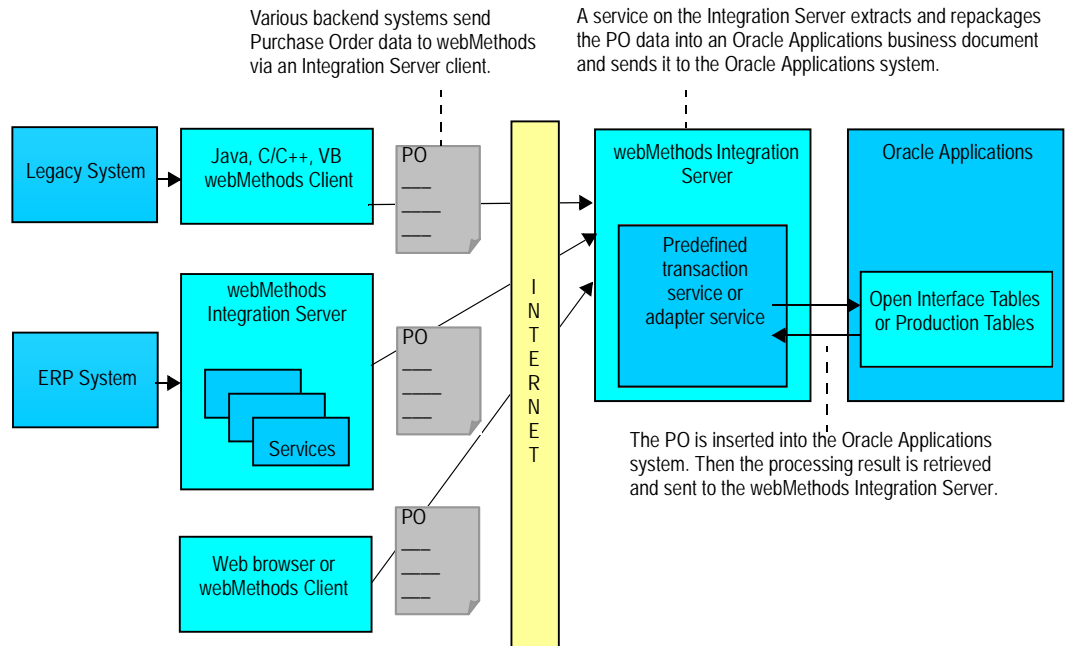
The adapter’s polling notification template is available in the webMethods Developer. For a detailed description of polling notifications, see [“Adapter Notifications” on page 20](#).

- **Oracle Thin JDBC Driver.** Enables the Oracle Applications Adapter to access the data in an Oracle Applications system. For more information about the JDBC driver, see [“Setting Up Your Adapter’s Environment” on page 35](#).
- **webMethods Repository.** The Repository contains the adapter’s transaction definition information. It also maintains the status of services running in the cluster.



Note: With the Oracle Applications Adapter version 6.0, transaction definitions used by adapter services you have created must reside in the Repository. In earlier versions of the adapter, once you generated an adapter service, you could remove its underlying transaction definition from the Repository. However, if you are upgrading to the Oracle Applications Adapter version 6.0 you must reestablish the contents of the Repository by importing the transaction definitions for your adapter services. This is necessary for your adapter services and adapter notifications to function properly. For more information, see [“Importing Transaction Definitions” on page 46](#).

The following diagram illustrates a typical business-process integration involving the Oracle Applications Adapter.



Package Management

The Oracle Applications Adapter is provided as a package called WmOAAAdapter. You manage the adapter package the same way you manage any package on the Integration Server. There are several considerations regarding how you set up and manage your packages on the Integration Server, such as those described in the following list.

- Create user-defined packages for your connections, adapter services, and adapter notifications. See [“Oracle Applications Adapter Package Management” on page 24](#) for details.
- Understand how package dependencies work so you make the best decisions regarding how you manage your connections, adapter services, and adapter notifications. See [“Setting Package Dependencies” on page 27](#) for details.
- Control which development groups have access to which adapter services and adapter notifications. See [“Group Access Control” on page 27](#) for details.
- Understand how clustering, an advanced feature of the webMethods Integration Server, works to effectively manage your adapter services and adapter notifications. See [“Oracle Applications Adapter in a Clustered Environment” on page 28](#) for details.

- Enable and disable packages. See [“Enabling and Disabling Packages” on page 26](#) for details.
- Load, reload, and unload packages. See [“Loading, Reloading, and Unloading Packages” on page 26](#) for details.

Adapter Connections

The Oracle Applications Adapter connects to an Oracle Applications system using a JDBC driver and adapter connections.

Adapter connections contain sets of connection parameters, including the logon parameters that the adapter needs to connect to the Oracle Applications Server and the release of the Oracle Applications Server. Using the Integration Server Administrator, you configure Oracle Applications Adapter connections for use with adapter services and adapter notifications. You must configure an adapter connection before you can configure an adapter service or adapter notification.

If you are configuring your own adapter services, you configure one or more adapter connections at design time to use in integrations. To use the Oracle Applications Adapter you must first configure one or more adapter connections, depending on your environment. For example, if you have multiple installations of Oracle Applications, you can access each installation using separate adapter connections. In addition, if you have a test system and a production system, you can configure an adapter connection for each system.

If you are using the predefined transaction services, you do not need to configure a connection because one is included. The underlying transaction definitions used by the predefined transaction services have all been preconfigured to use the same connection, allowing the services to connect to only one Oracle Applications system at a time. You only need to edit the connection parameters to specify your database connection information. For more information about how connections are used with the predefined transaction services, see the *webMethods Oracle Applications Adapter Predefined Transaction Services Users's Guide* for the Oracle Applications version with which you are using the adapter.

For instructions for configuring, viewing, editing, enabling, and disabling Oracle Applications Adapter connections, see [Chapter 3, “Adapter Connections” on page 33](#). For information about setting user privileges, see the *webMethods Integration Server Administrator's Guide*.

Using the JDBC Driver to Connect to Oracle Applications

Oracle Applications Adapter connections access databases using the Oracle Thin JDBC driver.

The JDBC driver must be installed on the Integration Server machine and loaded when the adapter starts. See the *webMethods Oracle Applications Adapter Installation Guide* for instructions.

Transaction Management of Oracle Applications Adapter Connections

Oracle Applications Adapter connections support the `LOCAL_TRANSACTION` transaction type. With this transaction type, all of the operations on the same connection in one transaction boundary will be committed or rolled back together. A transaction boundary means the scope of the transaction, from the beginning to the end of a transaction. It can be in one adapter service, one flow service, one Java service, or several steps in a flow service. This transaction type maintains the integrity of your business documents.



Note: The adapter also provides a `NO_TRANSACTION` type; however, we strongly suggest that you do not use it. Using this type of configuration the connection automatically commits (Auto Commit) all database operations.

When you configure a connection, the transaction type that you choose determines the type of transaction management that the connection's operations use implicitly. Implicit transactions, which include the transactions types in the preceding table, are managed by the Integration Server's transaction manager.

You can also explicitly manage transactions using built-in services. See [Appendix A, "Built-In Transaction Management Services"](#) on [page 97](#) for information about and examples of explicitly managing transactions.

Connection Pools

The Integration Server includes a connection management service that dynamically manages connections and connection pools based on configuration settings that you specify for the connection. All adapter services use connection pooling.

A connection pool is a collection of connections with the same set of attributes. The Integration Server maintains connection pools in memory. Connection pools improve performance by enabling adapter services to reuse open connections instead of opening new connections.

Run-Time Behavior of Connection Pools

When you enable a connection, the Integration Server initializes the connection pool, creating the number of connection instances you specified in the connection's **Minimum Pool Size** field when you configured the connection. Whenever an adapter service needs a connection, the Integration Server provides a connection from the pool. If no connections are available in the pool, and the maximum pool size has not been reached, the server creates one or more new connections (according to the number specified in the **Pool Increment Size** field) and adds them to the connection pool. If the pool is full (as specified in **Maximum Pool Size** field), the requesting service will wait for the Integration Server to obtain a connection, up to the length of time specified in the **Block Timeout** field, until a

connection becomes available. Periodically, the Integration Server inspects the pool and removes inactive connections that have exceeded the expiration period that you specified in the `Expire Timeout` field.

If you are using versions of the Integration Server earlier than 6.1 and the connection pool initialization fails because of a network connection failure or some other type of exception, you must disable the connection, fix the problem, and re-enable the connection. When using Integration Server 6.1, however, you can enable the system to retry the initialization any number of times, at specified intervals. For information about configuring connections, see [Chapter 3, “Adapter Connections”](#) on page 33.

Built-In Services For Connections

Integration Server 6.1 provides built-in services that enable you to programmatically control connections. You can use them to enable and disable a connection, and to return usage statistics and the current state (enabled/disabled) and error status for a connection. These services are located in the WmART package, in the `pub.art.connection` folder.

There are two built-in services, `setAdapterServiceNodeConnection` and `setPollingNotificationNodeConnection`, that enable you to change the connection associated with an adapter service or notification respectively. See [“Changing the Connection Associated With an Adapter Service or Notification at Design Time”](#) on page 16.

For details, see the *webMethods Integration Server Built-In Services Reference*, which is available from the Help menu of the webMethods Developer.

Changing the Connection Associated With an Adapter Service or Notification at Design Time

When using Integration Server 6.0.1 SP2, you may *not* assign a different connection to an adapter service or notification after you configure it. (You may, however, change the parameters of the connection.) If you want to use a different connection, you must configure a new adapter service or notification that specifies this new connection. For example, this means that in order to use the same adapter service for multiple Oracle Applications instances, you must configure multiple, separate adapter services, one for each connection you use.

To solve this limitation, Integration Server 6.1 provides built-in services that you can use at design time to change the connection associated with an adapter service or notification. These built-in services are named `setAdapterServiceNodeConnection` and `setPollingNotificationNodeConnection` and are provided in the WmART package’s `pub.art.service` folder. Using these services, you can change the specific connection associated with an adapter service at design time so that you do not need to recreate adapter services or notifications.



Note: These built-in services can be run at design time only; do not use them within an Integration Server flow or Java service. You must run this service directly from the Developer by selecting the service in the Developer and running it.

For details, see the *webMethods Integration Server Built-In Services Reference*, which is available from the Help menu of the webMethods Developer.

Other built-in services enable you to control connections; for more information, see [“Built-In Services For Connections” on page 16](#).

Transaction Definitions

Transaction definitions define the processing that the adapter services perform in Oracle Applications systems. Transaction definitions can map to open interface tables, stored procedures, views, and raw SQL statements.

Sample transaction definitions are available that you can use to configure your own adapter services. You can modify the sample transaction definitions or create new transaction definitions customized for your needs. For more information about the sample transaction definitions, see the *webMethods Oracle Applications Adapter Sample Transaction Definitions User's Guide*.

If you are using the predefined transaction services, you do not need to do anything with the transaction definitions. However, if you want to customize any of the transaction definitions from which the services were partially created, or if you want to configure new transaction definitions, you can do so using the adapter's Administrator screens. For example, you might want to customize a transaction definition for a receive service if the service delivers more business document fields than you need. For more information about the transaction definitions used with the predefined transaction services, see the *webMethods Oracle Applications Adapter Predefined Transaction Services User's Guide* for the Oracle Applications version with which you are using the adapter.

For information about importing, exporting, configuring, and managing transaction definitions, see [Chapter 4, “Transaction Definitions” on page 45](#).

Adapter Services

Adapter services enable an Integration Server to initiate operations on Oracle Applications systems. Adapter services use transaction definitions to determine the processing to perform on an Oracle Applications system.

The Oracle Applications Adapter enables you to configure customized adapter services that you can use to integrate with Oracle Applications systems. You configure adapter services using adapter service templates, which are provided with the Oracle Applications Adapter. For more information about the templates, see [“Adapter Service Templates” on page 18](#).

You will need to configure your own adapter services if:

- You are using the predefined transaction services but you have a highly customized Oracle Applications system and some of the predefined transaction services do not meet your needs, or
- You want to continue to use the adapter services that were created with versions of the Oracle Applications Adapter earlier than 6.0.

You can create a webMethods client that invokes the adapter service, or you can integrate the adapter service into an existing flow service. For example, if you want to code a webMethods client that creates a customer order from a purchase order, code the webMethods client to invoke the service that executes the `OEOrderImport` transaction of Oracle Order Management.

Adapter Service Templates

You use adapter service templates to configure adapter services. Each adapter service template represents a specific Oracle Applications operation. For example, the `Insert SQL` template enables you to insert business documents into the Oracle Applications open interface tables in one operation.

Adapter service templates contain all of the code that is necessary to perform an operation on an Oracle Applications system, but without the Oracle Applications data specifications. You provide these specifications through transaction definitions when you configure an adapter service.

Configuring a new adapter service from an adapter service template is straightforward. Using the webMethods Developer, you assign the service an adapter connection, select the adapter service template, and supply your transaction definition.

After you configure an adapter service, you can incorporate it into a flow service or Java service to interact with an Oracle Applications system. For example, you can create a flow service that collects customer order information from another one of your backend systems and then within that service you can call an `Insert SQL` service to insert the customer order information into your Oracle Applications system.

The Oracle Applications Adapter provides the following adapter service templates for adapter services:

Adapter Service Template	Description
Insert SQL	<p>This type of service is used to insert business documents into Oracle Applications using the open interface tables names specified in the IS-to-Oracle Applications transaction definitions.</p> <p>This service may be executed with an Error service (using the Simple SQL service template) to verify the status of the import performed by the concurrent process.</p> <p>See “Configuring Insert SQL Services” on page 64 for instructions.</p>
Compound Select SQL	<p>This type of service is used to query an Oracle Applications system and retrieve data based on the query conditions (specified in either a SQL tree or stored procedure).</p> <p>This service may be executed with an Acknowledge service (using the Simple SQL service template) to notify Oracle Applications that the document has been processed. This prevents the document from being retrieved multiple times.</p> <p>See “Configuring Compound Select SQL Services” on page 65 for instructions.</p>
Simple SQL	<p>Use this template to create Acknowledge services or Error services:</p> <ul style="list-style-type: none"> ■ When used with a Compound Select SQL service, Acknowledge services maintain an accurate status of the transactions within the Oracle system. You create an Acknowledge service from an Oracle Applications-to-IS transaction definition that either calls a stored procedure or executes a SQL UPDATE statement to tell the Oracle Applications instance that the transaction has been processed. ■ When used with an Insert SQL service, Error services query Oracle Applications for errors that might have occurred during the import process. You create an Error service from an IS-to-Oracle Applications transaction definition that is based on a stored procedure, view, or SQL SELECT statement. The service receives the query parameters for the SQL statement via the service specification. <p>See “Configuring Simple SQL Services” on page 66 for instructions.</p>

Predefined Transaction Services

Predefined transaction services are complete, ready-to-use flow services that simplify the process of integrating with Oracle Applications systems. Available as an add-on to the Oracle Applications Adapter, predefined transaction services are available for different versions of Oracle Applications.

In most cases, you can use these services as they are supplied. Some cases may require you to configure a connection to specify your Oracle Applications system connection information. However, if necessary, you can use the adapter to modify the services or their underlying transaction definitions.

See the *webMethods Oracle Applications Adapter Predefined Transaction Services User's Guide* for the Oracle Applications version with which you are using the adapter for a detailed description of the components involved in the predefined transaction services. The guides also include information about the transaction definitions used by the predefined transaction services, including where to locate them and how to import them into your webMethods system.

Adapter Notifications

An adapter notification enables the Oracle Applications Adapter to retrieve data from an Oracle Applications system at regular intervals and publish a document. The notification retrieves the data based on query conditions specified in either a SQL Tree or a stored procedure.

You must create a trigger on the Integration Server for the publishable document associated with the notification. This trigger identifies the receiver of the published document. For more information about creating triggers and working with publishable documents, see the *Publish-Subscribe Developer's Guide*.



Note: Adapter notifications do not use Acknowledge services. Without an Acknowledge service you may not be able to guarantee that the Oracle Applications event is only retrieved once. If you need to guarantee that the Oracle Applications event is retrieved only once, you should create the notification using a stored procedure, and that stored procedure should update the record in Oracle Applications to indicate that it is being processed.

Any exception produced by the notification is logged in the Integration Server Error Audit Log.

Use the webMethods Developer to configure, edit, and delete polling notifications. All notifications are configured from a notification template and require a configured connection.

Using the Integration Server Administrator you can display all polling notifications that have been configured for the Oracle Applications Adapter. You can also enable and

notifications and schedule them for execution. A notification can also be disabled; however, this is separate from the containing Integration Server package being disabled. If the Integration Server package is disabled, the notification is suspended, but the notification is not disabled. When the package containing the notification is enabled the scheduled execution resumes. If the notification is disabled it will not be executed until it is enabled.

For instructions about how to configure and manage notifications, see [Chapter 6, “Adapter Notifications”](#) on [page 73](#).

webMethods Manager Support for the Oracle Applications Adapter

The webMethods Manager Server is a systems management facility based on the Open Management Interface (OMI). It automatically manages the Oracle Applications Adapter components if the Manager Server manages the Integration Server on which the Oracle Applications Adapter is installed. The managed Oracle Applications Adapter components are:




- Adapter connections
- Adapter services
- Polling notifications

See the *webMethods Integrated Systems Management Programmer’s Guide* for information about the object interfaces available to the Oracle Applications Adapter. These object interfaces are the specific attributes and operations that you can perform to manage the Oracle Applications Adapter components.

Viewing Different Perspectives of the webMethods Developer

The webMethods Developer version 6.1 enables you to view its interface in three different perspectives: Edit, Test, and Details.

These perspectives enable you to display only those areas that are relevant to your current task. You can switch perspectives by using the following icons, which are located on the top right of the Developer window. Alternatively, you can access them from the **Window** menu.

Icon	Perspective Name	Description
	Edit Perspective	Displays all Developer areas but minimizes the Results panel. Use for opening and editing element.
	Test Perspective	Hides the Navigation and Recent Elements panels and maximizes the editor and Results panel. Use for testing and debugging an adapter service where you want to view the results of the service's execution, its inputs and outputs, and its pipeline variables.
	Details Perspective	Hides the Navigation and Recent Elements panels and minimizes the Results panel. Use when you want to see element details.

Package Management

■ Overview	24
■ Oracle Applications Adapter Package Management	24
■ Group Access Control	27
■ Oracle Applications Adapter in a Clustered Environment	28

Overview

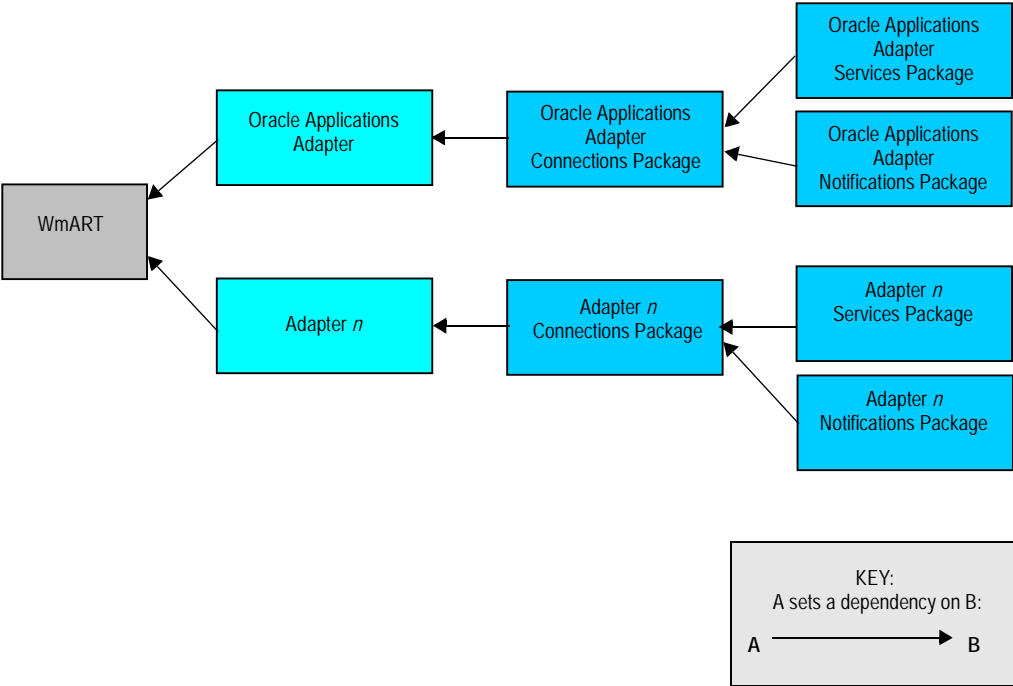
The following sections describe how to set up and manage your Oracle Applications Adapter packages, set up Access Control Lists (ACL), and use the adapter in a clustered environment.

Oracle Applications Adapter Package Management

The Oracle Applications Adapter is provided as a package called WmOAAadapter. You manage the WmOAAadapter package as you would manage any package on the Integration Server.

When you configure connections, adapter services, and adapter notification, define them in user-defined packages rather than in the WmOAAadapter package. Doing so will allow you to manage the package more easily.

As you create user-defined packages in which to store connections, adapter services, and adapter notifications, use the package management functionality provided in the Developer and set the user-defined packages to have a dependency on the WmOAAadapter package. That way, when the WmOAAadapter package loads or reloads, the user-defined packages load automatically. See the following diagram:



Package management tasks include:

- Setting package dependencies (see [“Package Dependency Requirements and Guidelines” on page 25](#)).
- [“Enabling and Disabling Packages” on page 26](#).
- [“Loading, Reloading, and Unloading Packages” on page 26](#).

Package Dependency Requirements and Guidelines

Following are dependency requirements and guidelines for user-defined packages. For instructions for setting package dependencies, see the *webMethods Developer User’s Guide*.

- A user-defined package must have a dependency on its associated adapter package, WmOAAAdapter. (The WmOAAAdapter package has a dependency on the WmART package.)

These dependencies ensure that at startup the webMethods Integration Server automatically loads or reloads all packages in the proper order: the WmART package first, the adapter package next, and the user-defined package(s) last. The WmART package is automatically installed when you install the Integration Server. You should not need to manually reload the WmART package.
- If the connections and adapter services of an adapter are defined in different packages, then:
 - A package that contains the connection(s) must depend on the adapter package.
 - Packages that contain adapter services must depend on their associated connection package.
- Keep connections for different adapters in separate packages so that you do not create interdependencies between adapters. If a package contains connections for two different adapters, and you reload one of the adapter packages, the connections for both adapters will reload automatically.
- The Integration Server will not allow you to enable a package if it has a dependency on another package that is disabled. That is, before you can enable your package, you must enable all packages on which your package depends. For information about enabling packages, see [“Enabling and Disabling Packages” on page 26](#).
- The Integration Server *will* allow you to disable a package even if another package that is enabled has a dependency on it. Therefore, you must manually disable any user-defined packages that have a dependency on the adapter package before you disable the adapter package.

Enabling and Disabling Packages

The WmOAAadapter package is enabled by default. You must manually enable all other packages.

To prevent the Integration Server from loading a particular package at startup, you must manually disable that package, using the Management window of the Integration Server Administrator. To do this, click **Yes** in the **Enabled** column for the package; the **Yes** changes to **No** (disabled). Disabled packages are not listed in the webMethods Developer. A disabled adapter will remain disabled until you explicitly enable it using the Integration Server Administrator.

To re-enable a package, click **No** to change it to **Yes** (enabled).



Note: Enabling an adapter package will *not* cause its associated user-defined package(s) to be reloaded. For information about reloading packages, see [“Loading, Reloading, and Unloading Packages” on page 26](#).



Important! Before you manually enable a user-defined package, *you must first enable its associated adapter package* (WmOAAadapter). Similarly, if your adapter has multiple user-defined packages, and you want to disable some of them, disable the adapter package first. Otherwise, errors will be issued when you try to access the remaining enabled user-defined packages.

Loading, Reloading, and Unloading Packages

Recall that if user-defined packages are properly configured with a dependency on the adapter package (as described in [“Package Dependency Requirements and Guidelines” on page 25](#)), at startup the Integration Server automatically loads or reloads all packages in the proper order: the WmART package first, the adapter package next, and the node package(s) last. You should not need to manually reload the WmART package.

Reloading Packages Manually

Reloading a user-defined package will *not* cause its associated adapter package to be reloaded. You can reload adapter packages and user-defined packages from either the Integration Server Administrator (by clicking the **Reload** icon on the Management window) or from the webMethods Developer (by right-clicking the package and selecting the **Reload Package** option from the menu).

Unloading Packages

At shutdown, the Integration Server unloads packages in the reverse order in which it loaded them: it unloads the node package(s) first, the adapter package next, and the WmART package last (assuming the dependencies are correct).

Setting Package Dependencies

You set package dependencies if a given package needs services in another package to load before it can load. For example, any packages you create for Oracle Applications Adapter services or notifications should identify the webMethods Oracle Applications Adapter package (WmOAAAdapter) as a package dependency because they require services in the WmOAAAdapter to load first. Use the following guidelines:

- Set package dependencies from the adapter service or notification package to the package containing the connection if you configure a connection in one package and the adapter services or notifications in another package. That is, the package that contains the connection should load before the adapter service or notification package.

When you set this package dependency, it ensures that if someone disables the connection package and then re-enables it, the adapter services or notifications will reload correctly.

- If both the connection and adapter services or notifications are in the same package, then no dependencies need to be set.
- In general, packages containing connections should have a dependency set to the adapter package itself. That is, the adapter service or notification package should depend on the adapter connection package, which should depend on the adapter package. Similarly, if the adapter services or notifications are in the same package as the connections, the only dependency that you need to set is between the adapter connection package and the adapter package.

For more information about setting package dependencies, see the *webMethods Developer User's Guide*.

Group Access Control

To control which development group has access to which adapter services, use access control lists (ACLs). You can use ACLs to prevent one development group from inadvertently updating the work of another group, or to allow or deny access to services that are restricted to one group but not to others.

For general information about assigning and managing ACLs, see the *webMethods Developer User's Guide*.

Oracle Applications Adapter in a Clustered Environment

What is webMethods Integration Server Clustering?

Clustering is an advanced feature that substantially extends the reliability, availability, and scalability of the webMethods Integration Server.

The clustering feature uses a shared *cluster store* to hold Integration Server state information and utilization metrics for use in load balancing and automatic failover support. Because this activity is transparent to the client, clustering makes multiple servers look and behave as one.

With clustering, you get the following benefits:

Load balancing. This feature, provided automatically when you set up a clustered environment, allows you to spread the workload over several servers, thus improving performance and scalability.

Failover support. Clustering enables you to avoid a single point of failure. If a server cannot handle a request, or becomes unavailable, the request is automatically redirected to another server in the cluster.



Note: Integration Server clustering redirects HTTP and HTTPS requests, but does not redirect FTP or SMTP requests.

Scalability. You can increase your capacity even further by adding new machines and the Integration Server to the cluster.

For details on Integration Server clustering, see the *webMethods IS Clustering Guide*.

Configuring the Oracle Applications Adapter

When you use the Oracle Applications Adapter to configure adapter services, you must:

- Ensure that each Integration Server in the cluster contains an identical set of packages.
- Disable the redirection capability for certain predefined administrative services.

Replicating Packages to webMethods Integration Servers

Every Integration Server in the cluster should contain an identical set of packages that you define using the Oracle Applications Adapter; that is, you should replicate the Oracle Applications Adapter services and the connections they use. You do not replicate the polling notifications.

To ensure consistency, we recommend that you create all packages on one server, and replicate them to the other servers. If you allow different servers to contain different

services, you might not derive the full benefits of clustering. For example, if a client requests a service that resides in only one server, and that server is unavailable, the request cannot be successfully redirected to another server.

For information about replicating packages, see the chapter on managing packages in the *webMethods Administrator's Guide*.

Disabling the Redirection of Administrative Services

As mentioned in [“What is webMethods Integration Server Clustering?” on page 28](#), a server that cannot handle a client's service request can automatically redirect the request to another server in the cluster. However, the Oracle Applications Adapter uses certain predefined administrative services that you should not allow to be redirected. These services are used internally when you configure the adapter. If you allow these services to be redirected, your configuration specifications might be saved on multiple servers, which is an undesirable result. For example, if you configure two Oracle Applications Adapter services, one might be stored on one server, while the other one might be stored on another server. Remember that all adapter services must reside on all webMethods Integration Servers in the cluster.

To disable the redirection of administrative services

- 1 Shut down the Administrator. See the *webMethods Administrator's Guide* for the procedure to do this.
- 2 Edit the following file:
`IntegrationServer_Directory\config\redir.cnf`
- 3 Add the following line to the file:
`<value name="wm.art">false</value>`
- 4 Save the file and restart the Integration Server.

Clustering Considerations and Requirements



Note: The following sections assume that you have already configured the webMethods Integration Server cluster. For details about webMethods clustering, see the *webMethods Integration Server Clustering Guide*.

The following considerations and requirements apply to the Oracle Applications Adapter in a clustered environment.

Requirements for Each Integration Server in a Cluster

The following table describes the requirements of each Integration Server in a given cluster:

All Integration Servers in a given cluster must have identical...	For Example...
Integration Server versions	One Integration Server in the cluster cannot be version 6.0.2 and another Integration Server in the cluster be version 6.0.
Adapter packages	All adapter packages on one Integration Server should be replicated to all other Integration Servers in the cluster.
Adapter versions	One Integration Server in the cluster cannot have Oracle Applications Adapter version 6.0.2 and another Integration Server in the cluster have Oracle Applications Adapter version 6.0.
Adapter connections	<p>If you configure a connection to the database, this connection must appear on all servers in the cluster so that any Integration Server in the cluster can handle a given request identically.</p> <p>If you plan to use connection pools in a clustered environment, see “Considerations When Configuring Connections with Connection Pooling Enabled” on page 31.</p>
Adapter services	<p>If you configure a specific Insert adapter service, this same adapter service must appear on all servers in the cluster so that any Integration Server in the cluster can handle the request identically.</p> <p>If you allow different Integration Servers to contain different services, you might not derive the full benefits of clustering. For example, if a client requests a service that resides on only one server, and that server is unavailable, the request cannot be successfully redirected to another server.</p>

See [“Replicating Packages to webMethods Integration Servers” on page 28](#) for information about replicating adapter packages, connections, and adapter services across multiple Integration Servers in a cluster.

Considerations When Installing Oracle Applications Adapter Packages

For each Integration Server in the cluster, use the standard Oracle Applications Adapter installation procedures for each machine, as described in the *webMethods Oracle Applications Adapter Installation Guide*.

Considerations When Configuring Connections with Connection Pooling Enabled

When you configure a connection that uses connection pools in a clustered environment, be sure that you do not exceed the total number of connections that can be opened simultaneously for that database.

For example, if you have a cluster of two Integration Servers with a connection configured to a database that supports a maximum of 100 connections opened simultaneously, the total number of connections possible at one time must not exceed 100. This means that you cannot configure a connection with an initial pool size of 100 and replicate the connection to both servers, because there could be possibly a total of 200 connections opened simultaneously to this database.

In another example, consider a connection configured with an initial pool size of 10 and a maximum pool size of 100. If you replicate this connection across a cluster with two Integration Servers, it is possible for the connection pool size on both servers to exceed the maximum number of database connections that can be open at one time.

For information about configuring connections for the Oracle Applications Adapter, see [“Configuring Adapter Connections” on page 35](#).

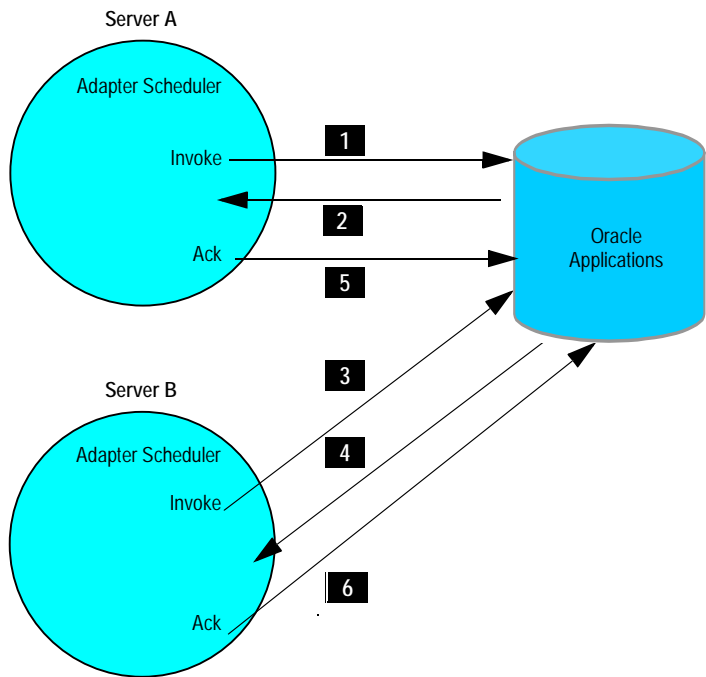
For more general information about connection pools, see the *webMethods Integration Server Administrator’s Guide*.

Scheduling Service Invocations in a Clustered Environment

Services configured from the Oracle Applications Adapter can be invoked within a flow that is scheduled from the webMethods scheduler. This is a useful process to invoke Oracle Applications-to-IS query services and IS-to-Oracle Applications Error SQL services.

In a clustered environment, it is important that you do not schedule the same query service using the same query parameters on multiple cluster servers. Doing this can produce duplicate data. This precaution applies only to the adapter services, and you will need to keep this in mind if you configure your own services. The predefined transaction services were designed to prevent multiple retrieval of a single document.

For example, assume you have two cluster servers: *Server A* and *Server B*. You schedule a *PurchaseOrderNew* service to be invoked every 15 minutes. This service is configured to retrieve all new purchase orders for *CompanyA*. Assume that you also schedule the same service on *Server B*. Because the scheduler is not aware of clustering, both services might execute simultaneously, which would cause duplicate purchase orders to be retrieved for *CompanyA*. The following figure illustrates this concept.



Step	Description
1	Scheduler A invokes a service to retrieve new purchase orders for <i>CompanyA</i> .
2	Server A begins processing the new purchase orders.
3	Scheduler B invokes a service to retrieve the new purchase orders.
4	Server B receives purchase orders already retrieved by Server A.
5	Server A acknowledges its purchase orders.
6	Server B acknowledges its purchase orders.

If you need to schedule PurchaseOrderNew, PurchaseOrderChange, InvoiceOutbound, or AdvancedShipNotice services on multiple servers, you must ensure that the query parameters on each server will return unique records because Oracle Applications will return the same records for the above transactions until they are acknowledged. The above example would work if *ServerA* was pulling new purchase orders for *CompanyA*, while *ServerB* was pulling new purchase orders for *CompanyB*. This must also be considered for any custom transaction that requires an acknowledgement.


Adapter Connections

■ Overview	34
■ Before Configuring or Managing Adapter Connections	34
■ Setting Up Your Adapter's Environment	35
■ Configuring Adapter Connections	35
■ Viewing Adapter Connection Parameters	40
■ Editing Adapter Connections	40
■ Copying Adapter Connections	41
■ Deleting Adapter Connections	42
■ Enabling Adapter Connections	43
■ Disabling Adapter Connections	43

Overview

The following sections provide instructions for configuring and managing Oracle Applications Adapter connections. For more information about how connections work, see [“Adapter Connections” on page 14](#).

Before Configuring or Managing Adapter Connections

 To prepare to configure or manage adapter connections


- 1 Install the webMethods Integration Server and the Oracle Applications Adapter on the same machine. See the *webMethods Oracle Applications Adapter Installation Guide* for details.
- 2 Make sure that you have webMethods administrator privileges so that you can access the Oracle Applications Adapter administrative screens. See the *webMethods Integration Server Administrator's Guide* for information about setting user privileges.
- 3 Start the Integration Server and the Integration Server Administrator, if they are not already running.
- 4 Using the Administrator, make sure the WmOAAdapter package is enabled. See [“Enabling and Disabling Packages” on page 26](#) for instructions.
- 5 Using the webMethods Developer, create a user-defined package for each connection you plan to configure. See [“Oracle Applications Adapter Package Management” on page 24](#) for more information about managing packages for the adapter.

Setting Up Your Adapter's Environment

The Oracle Applications Adapter requires a JDBC driver (typically `classes12.zip`) that has the same version as your Oracle database.

The webMethods Installer enables you to either copy the file at install time, or to copy the file later. (See the *webMethods Oracle Applications Adapter Installation Guide* for details.) If you did not perform that step at install time, you will need to do it before configuring an adapter connection.

Ensure that the classes are correct for the version of the Oracle DBMS you are using.

 To provide access to the Oracle thin driver classes

- 1 Locate the file `classes12.zip`. Typically it is available on the Oracle server.
- 2 Copy the file to the following directory:

Integration Server_directory\packages\WmOAAAdapter\code\jars


Configuring Adapter Connections

When you configure Oracle Applications Adapter connections, you create a new connection and specify information that the Integration Server uses to connect and log into an Oracle Applications server.

You configure Oracle Applications Adapter connections using the Integration Server Administrator.



Note: If you are using predefined transaction services, they use a specific preconfigured connection. You only need to edit the connection and specify the database connection information specific to your system. See [“Editing Adapter Connections” on page 40](#) for instructions. For more information about using connections with predefined transaction services, see the *webMethods Oracle Applications Adapter Predefined Transaction Services User's Guide* for the Oracle Applications version with which you are using the adapter.

 To configure a connection

- 1 In the **Adapters** menu in the Administrator's navigation area, click **Oracle Apps Adapter**.
- 2 On the **Connections** screen, click **Configure New Connection**.
- 3 On the **Connection Types** screen, click **Oracle Apps Adapter Connection** to display the **Configure Connection Type** screen.

- 4 In the Oracle Apps Adapter section, provide values for the following fields.

Parameter	Description/Action
Package	<p>The package in which to configure the connection.</p> <p>You must create the package using the Developer before you can specify it for this parameter. For general information about creating packages, see the <i>webMethods Developer User's Guide</i>.</p> <hr/> <p>Note: Configure the connection in a user-defined package rather than in the adapter's package. See “Oracle Applications Adapter Package Management” on page 24 for other important considerations when creating packages for use with the Oracle Applications Adapter.</p> <hr/>
Folder Name	The folder in which to configure the connection.
Connection Name	The name you want to give the connection. Connection names cannot have spaces or use special characters reserved by the Integration Server or Developer. See the <i>webMethods Developer User's Guide</i> for more information about the use of special characters in package, folder, and element names.

- 5 In the Connection Properties section, provide values for the following fields:

Parameter	Description/Action
Oracle Apps Release version	The version of the Oracle Applications server to which you are connecting.
Database Server Host	The host name of the Oracle Applications database to which you are connecting.
Database Port Number	The port number of the Oracle Applications database.
Database Name	The name of the Oracle Applications database.
Database User Name	<p>The user name that the webMethods Integration Server must use to log into the Oracle Applications database. Typically the user name is APPS.</p> <p>The user name you specify must have access to the Oracle package that contains the Oracle Applications tables and the adapter database setup in the Oracle DBMS.</p>
Database Password	The password that the webMethods Integration Server must use to log into the Oracle Applications database.

Parameter	Description/Action
Retype Database Password	Confirms the password you specified in Database Password.
Local/XA Transaction Control	<p>The type of transaction support that the connection provides. Select the following transaction type:</p> <p>LOCAL_TRANSACTION: The connection does not automatically commit transactions. You can manually define the transactions, or the Integration Server's transaction manager will manage it for you. See Appendix A, "Built-In Transaction Management Services" on page 97 for instructions on managing transactions manually.</p>
DataSource Class Name	<p>The name of the JDBC driver's DataSource class name. Only the Oracle Thin JDBC Driver is supported:</p> <pre>oracle.jdbc.pool.OracleDataSource</pre>
Network protocol	<p>A standard Oracle Applications DataSource property to indicate the name of the network protocol that the connection will use when connecting to the database.</p> <p>Leave this parameter blank.</p>
Other properties	<p>Enables you to provide additional JDBC driver DataSource properties for the Oracle Thin JDBC Driver. The following property is supported:</p> <pre>driverType=thin</pre>

- 6 In the Connection Management Properties section, provide values for the following fields:

Parameter	Description/Action
Enable Connection Pooling	<p>Enables the connection to use connection pooling.</p> <p>See “Connection Pools” on page 15 for more information about connection pooling.</p> <hr/> <p>Note: If you plan to enable connection pooling in a clustered environment, consider the connection pool size. For details, see “Considerations When Configuring Connections with Connection Pooling Enabled” on page 31.</p> <hr/>
Minimum Pool Size	If connection pooling is enabled, this field specifies the number of connections to create when the connection is enabled. The adapter will keep open the number of connections you configure here regardless of whether these connections become idle.
Maximum Pool Size	If connection pooling is enabled, this field specifies the maximum number of connections that can exist at one time in the connection pool.
Pool Increment Size	If connection pooling is enabled, this field specifies the number of connections by which the pool will be incremented if connections are needed, up to the maximum pool size.
Block Timeout (msec)	<p>If connection pooling is enabled, this field specifies the number of milliseconds that the Integration Server will wait to obtain a connection with the database before it times out and returns an error.</p> <p>For example, you have a pool with Maximum Pool Size of 20. If you receive 30 simultaneous requests for a connection, 10 requests will be waiting for a connection from the pool. If you set the Block Timeout to 5000, the 10 requests will wait for a connection for 5 seconds before they time out and return an error. If the services using the connections require 10 seconds to complete and return connections to the pool, the pending requests will fail and return an error message stating that no connections are available.</p> <p>If you set the Block Timeout value too high, you may encounter problems during error conditions. If a request contains errors that delay the response, other requests will not be sent. This setting should be tuned in conjunction with the Maximum Pool Size to accommodate such bursts in processing.</p>

Parameter	Description/Action
Expire Timeout (msec)	<p>If connection pooling is enabled, this field specifies the number of milliseconds that an inactive connection can remain in the pool before it is closed and removed from the pool.</p> <p>The connection pool will remove inactive connections until the number of connections in the pool is equal to the Minimum Pool Size. The inactivity timer for a connection is reset when the connection is used by the adapter.</p> <p>If you set the Expire Timeout value too high, you may have a number of unused inactive connections in the pool. This consumes local memory and a connection on your backend resource. This could have an adverse effect if your resource has a limited number of connections.</p> <p>If you set the Expire Timeout value too low, performance could degrade because of the increased activity of creating and closing connections. This setting should be tuned in conjunction with the Minimum Pool Size to avoid excessive opening/closing of connections during normal processing.</p>
Startup Retry Count	(For Integration Server 6.1 only.) The number of times that the system should attempt to initialize the connection pool at startup if the initial attempt fails. The default is 0.
Startup Backoff Timeout (sec)	(For Integration Server 6.1 only.) The number of seconds that the system should wait between attempts to initialize the connection pool.


7 Click Save Connection.


The connection you configured appears on the adapter's Connections screen and in the Developer's Navigation Panel.

By default, when you configure a connection, it is not enabled. For information about enabling connections, see [“Enabling Adapter Connections” on page 43](#).

Viewing Adapter Connection Parameters


You can view a connection's parameters from the Integration Server Administrator or from the Developer.

 To view the parameters for a connection using the Administrator

- 1 In the **Adapters** menu in the Administrator's navigation area, click **Oracle Apps Adapter**.
- 2 From the navigation area, select **Connections**.
- 3 On the Connections screen, click the **View** icon  for the connection you want to see.

The View Connection screen displays the parameters for the connection. For descriptions of the connection parameters, see [“Configuring Adapter Connections” on page 35](#).

- 4 Click **Return to Oracle Apps Adapter Connections** to return to the main connections screen.

 To view the parameters for a connection using the Developer

- 1 From the Developer's navigation area, open the package and folder in which the connection is located.
- 2 Click the connection you want to view.

The parameters for the connection appear in the Connection Information tab. For descriptions of the connection parameters, see [“Configuring Adapter Connections” on page 35](#).

Editing Adapter Connections

You edit connections using the Administrator. If the login information for an Oracle Applications server changes, you must update your connection to reflect the changes.



Note: You can edit a connection only when it is disabled, as described in [“Disabling Adapter Connections” on page 43](#).

 To edit a connection

- 1 In the **Adapters** menu in the Administrator's navigation area, click **Oracle Apps Adapter**.
- 2 From the navigation area, select **Connections**.
- 3 Make sure that the connection is disabled before editing it. See [“Disabling Adapter Connections” on page 43](#) for instructions.

- 4 On the Connections screen, click the **Edit** icon  for the connection you want to edit.

The Edit Connection screen displays the current parameters for the connection. Update the connection's parameters by typing or selecting the values you want to specify.

For descriptions of the connection parameters, see [“Configuring Adapter Connections” on page 35](#).

- 5 Click **Save Changes** to save the connection and return to the Connections screen.
- 6 Enable the connection when you are ready to use it. See [“Enabling Adapter Connections” on page 43](#) for instructions.

Copying Adapter Connections

You copy connections using the Administrator. You can copy an existing Oracle Applications Adapter connection to configure a new connection with the same or similar connection properties without having to type all of the properties for the new connection.

To copy a connection

- 1 In the **Adapters** menu in the Administrator's navigation area, click **Oracle Apps Adapter**.
- 2 From the navigation area, select **Connections**.
- 3 On the Connections screen, click the **Copy** icon for the connection you want to copy.

The Copy Connection screen displays the current parameters for the connection you want to copy. Name the new connection, specify a Package Name and Folder Name, and edit any connection parameters as needed by typing or selecting the values you want to specify.



Note: When you copy a connection, the new connection does not save the password of the original connection. You must enter and then retype the password before you can save the new connection.

For descriptions of the connection parameters, see [“Configuring Adapter Connections” on page 35](#).

- 4 Click **Save Connection Copy** to save the connection and return to the Connections screen.

Deleting Adapter Connections

You delete connections using the Administrator. If you no longer want to use a particular Oracle Applications Adapter connection, you can delete it by following the instructions in this section.




Note: You can delete a connection only if it is disabled, as described in [“Disabling Adapter Connections” on page 43](#).



Important! If you delete an Oracle Applications Adapter connection, any adapter services or adapter notifications that are defined to use the connection will no longer work.

You cannot change which connection an adapter service uses after the service is configured; however, you can change the parameters for an existing connection. See [“Editing Adapter Connections” on page 40](#) for instructions.

To delete a connection

- 1 In the **Adapters** menu in the Administrator’s navigation area, click **Oracle Apps Adapter**.
- 2 On the **Connections** screen, make sure that the connection is disabled before deleting. See [“Disabling Adapter Connections” on page 43](#) for instructions.
- 3 On the **Connections** screen, click  for the connection you want to delete.
The Integration Server deletes the adapter connection.

Enabling Adapter Connections

You enable connections using the Administrator. An Oracle Applications Adapter connection must be enabled before you can configure an adapter service using the connection, or before an adapter service can use the connection at run time.



Note: When you reload a package that contains enabled connections, the connections will automatically be enabled when the package reloads. If the package contains connections that are disabled, they will remain disabled when the package reloads.



To enable a connection

- 1 In the **Adapters** menu in the Administrator's navigation area, click **Oracle Apps Adapter**.
- 2 On the **Connections** screen, click **No** in the **Enabled** column for the connection you want to enable.

The Integration Server Administrator enables the adapter connection and displays



and **Yes** in the **Enabled** column.

Disabling Adapter Connections

You disable connections using the Administrator. An Oracle Applications Adapter connection must be disabled before you can edit or delete it.



To disable a connection

- 1 In the **Adapters** menu in the Administrator's navigation area, click **Oracle Apps Adapter**.
- 2 On the **Connections** screen, click **Yes** in the **Enabled** column for the connection you want to disable.

The adapter connection becomes disabled and you see a **No** in the **Enabled** column.

Transaction Definitions

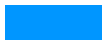
■ Overview	46
■ Before Working with Transaction Definitions	46
■ Importing Transaction Definitions	46
■ Exporting Transaction Definitions	48
■ Creating IS-to-Oracle Applications Transaction Definitions	50
■ Creating Oracle Applications-to-IS Transaction Definitions	51
■ Editing IS-to-Oracle Applications Transaction Definitions	52
■ Editing Oracle Applications-to-IS Transaction Definitions	53
■ Deleting Transaction Definitions	54
■ Working with SQL Trees in Oracle Applications-to-IS Transactions	55

Overview

The following sections provide instructions for importing and exporting Oracle Applications Adapter transaction definitions, and for configuring and managing those transaction definitions. For more information about how transaction definitions work, see [“Transaction Definitions” on page 17](#).

Additionally, this chapter provides a detailed explanation of SQL trees, and describes how to use them when configuring or updating transaction definitions in [“Working with SQL Trees in Oracle Applications-to-IS Transactions” on page 55](#).

Before Working with Transaction Definitions



To prepare to import, export, configure, or manage transaction definitions

- 1 Start the Integration Server and the Integration Server Administrator, if they are not already running.
- 2 Make sure that you have webMethods Administrator or Developer privileges so that you can access the Oracle Applications Adapter administrative or Developer screens. See the *webMethods Integration Server Administrator's Guide* for information about setting user privileges.
- 3 Using the Administrator, make sure the WmOAdapter package is enabled. See [“Enabling and Disabling Packages” on page 26](#) for instructions.

Importing Transaction Definitions

You can use the procedure in this section to import the following types of transaction definitions into the webMethods Repository:

- Sample transaction definitions

If you plan to use the sample transaction definitions to configure adapter services, you must first import the transaction definitions. The sample transaction definitions are located in the following directory:

IntegrationServer_Directory\packages\WmOAdapter\sampleTxns



Note: The sample transaction definitions are the same transaction definitions that were provided with the Oracle Applications Adapter version 2.0.

- Transaction definitions used by the predefined transaction services

If you plan to use the predefined transaction services, you do not have to import their associated transaction definitions. This is because the transaction definitions that were installed as part of the predefined transaction services are loaded automatically by the Integration Server at startup. For the location of the transaction definition files, see the *webMethods Oracle Applications Adapter Predefined Transaction Services User's Guide* for the Oracle Applications version with which you are using the adapter.

- Transaction definitions that have previously been exported to files
- Transaction definitions that have been modified or custom built

After you import transaction definitions, they are stored in the webMethods Repository.

You must make sure that the files you want to import are located in the following directory:

IntegrationServer_Directory\packages\WmOAAAdapter\exchange



Note: Any .txp files in the \exchange directory are automatically imported when the Integration Server starts or when the WmOAAAdapter package is reloaded.

Transaction definition files must use the .txp file extension.

You can import transaction definition files in the \exchange folder from the Integration Server Administrator or from the Developer. The Administrator lets you import one transaction definition file at a time; using the Developer you can import all transaction definition files.

To import transaction definitions one at a time using the Administrator

- 1 In the **Adapters** menu in the Administrator's navigation area, click **Oracle Apps Adapter**.
- 2 On the **Connections** screen, in the **Oracle Apps Adapter** menu in the navigation area, click **Transaction Mgmt.a**
- 3 Click **Transaction Import**.
- 4 Select a file from the **Import Files** list.

If no files are listed, ensure that the transaction file to import is in the \exchange directory.


- 5 From the **Import to Oracle Applications Release** list, select the release of Oracle Applications for which you are importing transaction definitions. The list defaults to the release from which the transaction definitions were exported.
- 6 Select the transactions you want to import, as follows:

For this parameter...	Specify...
Import this transaction	Whether you want to import the transaction. Click Set to select all transactions to import. Click Clear to de-select all transactions selected for import.
Import Name	The name you want the transaction to have in your system. The default is the name as it is specified in the import file.
Overwrite Existing	Whether you want to overwrite transactions that are named the same as the import name. To overwrite all transactions, click Set . To de-select all transactions, click Clear .

7 Click **Import**. The adapter displays the **Oracle Applications Transaction Import Results** screen, which lists the status of the transaction definition you imported.



To import all transaction definitions using the Developer

- 1 Start the Developer if it is not already running. See the *webMethods Administrator's Guide* if you need the procedure for this step.
- 2 From the Developer's navigation area, open the WmOAAadapter package and folder in which the connection is located. Typically this is the **pub>adapter>wmoa** folder.
- 3 Double-click **ImportAllTransactions**.
- 4 Click the **Run** icon . Examine the Results pane to see the import status.

Exporting Transaction Definitions

You can export transaction definitions to files to back them up or to prepare to transfer them to a different installation of the Oracle Applications Adapter. You export transaction definitions from the webMethods Repository. Exported files are saved to the *IntegrationServer_Directory\packages\WmOAAadapter\exchange\export* directory.



To export transaction definitions

- 1 In the **Adapters** menu in the Administrator's navigation area, click **Oracle Apps Adapter**.
- 2 In the **Oracle Apps Adapter** menu in the navigation area, click **Transaction Mgmt**.
- 3 Click **Transaction Export**.

- 4 Select the release of the Oracle Applications transactions you want to export.

The Oracle Applications Adapter displays the **Oracle Applications Transaction Export** screen, which lists all available transaction definitions for the release of Oracle Applications you selected. Use this screen to specify which transaction definitions you want to export.


- 5 Provide information about the export file, as follows:

<u>For this parameter...</u>	<u>Specify...</u>
Oracle Apps Release	The release of Oracle Applications from which you want to export transaction definitions.
Export File	The name of the export file. If the file does not currently exist, the adapter will create it for you. You do not have to specify the .txp file extension; it is supplied for you automatically.
Overwrite Existing File	Whether to overwrite an existing file if there is already a file with the name you specified in Export File .
Create Host	The machine on which the Integration Server is running. If you want to change the name of the host for security purposes, type a new name in the field.
Description	A description to identify the contents of the file.

- 6 Select the transactions you want to export. To select all transactions to export, click **Set**. To de-select all transactions selected for export, click **Clear**.
- 7 Click **Create File**. The export file is created in the *IntegrationServer_Directory\packages\WmOAAadapter\exchange\export* directory.
- 8 When you are ready to import the exported transaction definitions, move the export file that is in the *\export* directory to the *\exchange* directory.

Creating IS-to-Oracle Applications Transaction Definitions

Perform the following procedure to create transaction definitions that insert data from the webMethods Integration Server into Oracle Applications tables.

 To create a new IS-to-Oracle transaction definition

- 1 In the **Adapters** menu in the Administrator's navigation area, click **Oracle Apps Adapter**.
- 2 In the **Oracle Apps Adapter** menu in the navigation area, click **Transaction Mgmt**.
- 3 Click **Transaction Configuration**.
- 4 Select the Oracle Applications release from the list.
- 5 Select **IS-to-Oracle** in the **Transaction Type** field if it is not currently selected.
- 6 Click **Add**. The adapter displays the **New Transaction** screen.
- 7 Specify a name for the transaction in the **Transaction Name** field.
- 8 Click **Submit**.
- 9 Enter the name of an open interface table in the **Table Name** field. If you want to specify that this table is required, check the **Required** option. Then click **Submit**.

Repeat this step for each table you want the transaction to have access to. If you want to add a new table but a table is currently selected, click **Cancel** to de-select the current table.

Each table added appears in the **Current Tables** area. If you want to delete a table from the list, select it and then click **Delete Table**.

- 10 To specify SQL to check for errors that occurred during the import process, click **Edit Error SQL**. The adapter displays the **Error Service SQL for *transaction*** screen. Add the SQL in the **Error SQL** field, and then click **Submit**.

To expose an SQL input parameter as a service input in the Developer, enclose the input parameter within an ampersand (&) and an asterisk (*). For example, &varname*. Stored procedure input parameters are exposed as service inputs in the Developer.

Creating Oracle Applications-to-IS Transaction Definitions

Perform the following procedure to create transaction definitions that select data from Oracle Applications and delivers it to the webMethods Integration Server.

See “[Working with SQL Trees in Oracle Applications-to-IS Transactions](#)” on page 55 for information about how SQL trees are defined in Oracle Applications-to-IS transaction definitions.



Note: You cannot combine stored procedures with SQL queries in an Oracle Applications-to-IS transaction definition. Use only stored procedures or SQL trees.

To create a new Oracle Applications-to-IS transaction definition

- 1 In the **Adapters** menu in the Administrator’s navigation area, click **Oracle Apps Adapter**.
- 2 In the **Oracle Apps Adapter** menu in the navigation area, click **Transaction Mgmt**.
- 3 Click **Transaction Configuration**.
- 4 Select the Oracle Applications release from the list.
- 5 Select **Oracle-to-IS** in the **Transaction Type** field.
- 6 Click **Add**. The adapter displays the **New Transaction** screen.
- 7 Specify a name for the transaction in the **Transaction Name** field.
- 8 Click **Submit**. The adapter displays the **Edit Transaction** screen.
- 9 Specify the top-level alias for the transaction, as follows:
 - a Enter the alias name in the **Alias Name** field.
 - b In the **SQL** field, enter or edit the SQL statement that performs the query, or enter the name of a stored procedure to execute:
 - To expose an SQL input parameter as a service input in the Developer, enclose the input parameter within an ampersand (&) and an asterisk (*). For example, `&varname*`. Stored procedure input parameters are exposed as service inputs in the Developer.
 - To call a stored procedure, enter the stored procedure name in the format `schema.package.procname`. If the stored procedure is not located in a package, you do not have to enter the schema and package names.
 - c Click **Submit**.

- 10 To specify a child for an alias, select the alias to which you want to add a child and click **Next Level**. If you want to add a new sibling alias but an alias is currently selected, click **Cancel** to de-select the current alias. Then specify the alias information, as follows:
 - a Enter the alias name in the **Alias Name** field.
 - b In the **SQL** field, enter or edit the SQL statement that performs the query.
 - c To specify the relationship of the current alias to its parent alias, enter it in the **Relations** field.

If you are creating an alias with a **SELECT** statement, **Relations** defines the **WHERE** clause and join parameters. Enter the values as follows:

```
COLUMN = PARENT_ALIAS.COLUMN AND COLUMN2 = PARENT_ALIAS.COLUMN2 ...
```
 - d Click **Submit**.
- 11 To add SQL to acknowledge the transaction, click **Edit Ack SQL**. The adapter displays the **Acknowledgement Service SQL for *transaction*** screen. Add the SQL in the **Acknowledgement SQL** field, and then click **Submit**.




Note: The purpose of an Acknowledgement SQL is to prevent the query from retrieving the same piece of data again.

- 12 Repeat the previous step for each alias you are adding.

Editing IS-to-Oracle Applications Transaction Definitions



Note: You should be familiar with the Oracle Applications data schema before editing transaction definitions. Please refer to your Oracle Applications technical documentation or consult your Oracle Applications Administrator before editing the transaction definitions.

 To edit transaction definitions

- 1 In the **Adapters** menu in the Administrator's navigation area, click **Oracle Apps Adapter**.
- 2 In the **Oracle Apps Adapter** menu in the navigation area, click **Transaction Mgmt**.
- 3 Click **Transaction Configuration**.
- 4 Select the Oracle Applications release from the list.
- 5 Select **IS-to-Oracle** in the **Transaction Type** field.

- 6 Select the transaction definition you want to edit, and then click **Edit**. The adapter displays the **Edit *transaction* Transaction** screen.
- 7 Update the transaction definition as follows:
 - To add a new table, enter the table name in the **Table Name** field, indicate whether the table is required, and then click **Submit**.
 - To delete a table, select the table from the list of current tables, and then click **Delete Table**.
 - To change the error log SQL, click **Edit Error SQL**. The adapter displays the **Error Service SQL For *transaction*** screen. Update the SQL in the **Error SQL** field, and then click **Submit**. The error log SQL is SQL code that checks for errors that occur during the import process.
- 8 When you are finished updating the transaction definition, click **Back**.



Important! If you change a transaction definition, you must refresh any services that were created from that transaction. Refreshing the services causes the Integration Server to update the information in its cache for the affected services. For more information about updating the information in the Developer's Navigation panel, see the *webMethods Developer User's Guide*.

Editing Oracle Applications-to-IS Transaction Definitions

See [“Working with SQL Trees in Oracle Applications-to-IS Transactions” on page 55](#) for information about how SQL trees are defined in Oracle Applications-to-IS transaction definitions.



Note: You should be familiar with the Oracle Applications data schema before editing transaction definitions. Please refer to your Oracle Applications technical documentation or consult your Oracle Applications Administrator before editing the transaction definitions.

To edit transaction definitions

- 1 In the **Adapters** menu in the Administrator's navigation area, click **Oracle Apps Adapter**.
- 2 In the **Oracle Apps Adapter** menu in the navigation area, click **Transaction Mgmt**.
- 3 Click **Transaction Configuration**.
- 4 Select the Oracle Applications release from the list.
- 5 Select **Oracle-to-IS** in the **Transaction Type** field.

- 6 Select the transaction definition you want to edit from the **Current Oracle-to-IS Transactions** list.
- 7 Click **Edit**.
- 8 Update the transaction definition as follows:
 - To update an existing alias, select the alias you want to edit from **Current SQL Tree** for: *transaction*. If you want to edit a child alias, drill down to the child alias by selecting the parent alias and clicking **Next Level**. Update the Alias Name, SQL, and Relations (if applicable) and then click **Submit**.
 - To delete an alias, select the alias from **Current SQL Tree** for: *transaction* and then click **Delete Alias**.
 - To add a new alias, select the alias to which you want to add a child alias and then click **Next Level**. Enter the Alias Name, SQL, and Relations information and then click **Submit**.
- 9 When you are finished updating the transaction definition, click **Previous Level** until you return to the **Transaction Configuration** screen.



Important! If you change a transaction, you must refresh any services that were created from that transaction. Refreshing the services causes the Integration Server to update the information in its cache for the affected services. For more information about updating the information in the Developer's Navigation panel, see the *webMethods Developer User's Guide*.

Deleting Transaction Definitions

If you no longer need an Oracle Applications transaction definition, you can delete it from the webMethods repository.



Important! Use caution when deleting transaction definitions. Deleting a transaction definition renders their associated adapter services useless.

To delete a transaction definition

- 1 In the **Adapters** menu in the Administrator's navigation area, click **Oracle Apps Adapter**.
- 2 In the **Oracle Apps Adapter** menu in the navigation area, click **Transaction Mgmt**.
- 3 Click **Transaction Configuration**.
- 4 From the **Oracle Apps Release** list, select the Oracle Applications release for the transaction you want to delete.

- 5 Select the appropriate Transaction Type.
- 6 Click on the transaction you want to delete.
- 7 Click Delete.

Working with SQL Trees in Oracle Applications-to-IS Transactions

The Oracle Applications Adapter uses SQL trees to build and run Oracle Applications-to-IS transaction definitions. SQL trees are hierarchical SQL statements that the adapter executes at runtime to retrieve data from an Oracle Applications system and to store the data in a hierarchical format that is easily accessible in integration flows.

SQL trees enable the adapter to collect data from separate tables and to place (join) the results in a hierarchical structure rather than in a standard result set. You must understand how the Oracle Applications Adapter uses SQL trees to define Oracle Applications-to-IS transaction definitions before you create or edit the transaction definitions.

SQL trees are made up of a series of SQL query calls called aliases. You can specify an alias's SQL query either as a SQL SELECT statement or as a stored procedure. Each SQL tree contains one top-level SQL statement and one or more child SQL statements. Child level SQL statements retrieve data from Oracle Applications using results from the top-level SQL statement as the query input parameters in their own SQL calls.

SQL SELECT Statement Syntax

When you specify a SQL SELECT statement for the top-level alias in the SQL tree, the SQL statement should include any WHERE statements or ORDER BY statements the SQL statement will use to issue the query. You will use the results of the top-level alias query as input parameters when you define the top-level's child aliases for the SQL tree.

Within the top-level alias's SQL statement you can define input variables that will be defined as service inputs when you configure a service using the transaction definition. To define a SQL statement's input as a variable, place the parameter between an ampersand (&) and an asterisk (*). You cannot define service input variables for non-top-level aliases.

When you specify a SQL statement for a child alias, define the SQL statement without defining the query's WHERE clause. You will define the input and join parameters for the child alias's SQL statement as the alias's *relations*. An alias's relations populate the child alias's input and join parameters using output values from alias's parent alias. You can also specify constant values as input parameters in a relation.

When you specify the query using a SQL SELECT statement, specify the alias's relations as name value pairs. Be sure to leave a space before and after the equals sign (=) in the name value pairs. For example:

```
COLUMN = PARENT_ALIAS.COLUMN AND COLUMN2 = PARENT_ALIAS.COLUMN2 ...
```

The results of a child alias' SQL query statement will be stored as a child section of the results record. The resulting child record will be named the alias name. The output names in the child record will be based on the column names returned from the alias's SELECT statement.

Stored Procedure Syntax

To specify a stored procedure in an alias, enter the stored procedure as follows:

```
schema.package.procname
```

If the stored procedure is not in a package, you do not have to specify the schema and the package.

To specify the relations for a child alias that issues a stored procedure, you can enter the stored procedure inputs as the parent alias's output values (if applicable), or you can enter the stored procedure inputs as constant values. Specify the inputs, separated by commas, in the order that the stored procedure requires them. You can also specify constant values as input parameters in a relation. Type parent columns in uppercase.

For example, specify the alias's relations as follows:

```
PARENT_ALIAS.COLUMN1, PARENT_ALIAS.COLUMN2 ...
```

The results of the stored procedure will be stored in the parent alias' result record. The outputs will be given the names of the stored procedures output parameters.



Note: You cannot add a child alias to an alias that issues a stored procedure.

SQL Tree Example

This section illustrates how to use SQL trees to retrieve data from an Oracle Applications system and to store it in a hierarchical format that is easy to access within flows.

Suppose you want a service to execute a transaction that retrieves customer names and phone numbers using a customer ID input parameter. For this example, the database contains the following tables and data:

TABLE1 looks like this:

ID	STATUS	NAME
001	N	John Smith

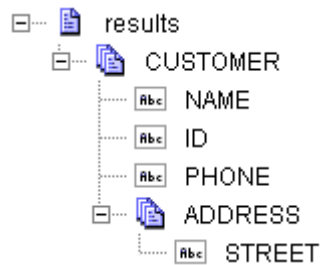
TABLE2 looks like this:

ID	PHONE
001	(888) 555-1212

TABLE3 looks like this:

ID	STREET
001	Main Street
001	Broadway Blvd.

Also, suppose you want the output from the transaction to be a results document in the following format:



Based on the customer's status, you want to use the customer's ID to retrieve the rest of the customer's information. To retrieve the customer's phone number, you want to use an existing stored procedure: GetCustomerPhone.

To define the transaction definition you will create a parent alias and two child aliases in the SQL tree, as follows:

Parent Level Alias

This alias retrieves the customer's ID and name based on the customer's status. You will use the ID field from this alias to query the database in the two child aliases.

Field	Value
Alias Name	CUSTOMERS
SQL	SELECT NAME, ID FROM TABLE1 WHERE STATUS = 'N' ORDER BY NAME

Child Alias 1

This child alias issues a stored procedure to gather phone number information for each customer. The stored procedure's input value is ID, and its output value is PHONE.

Field	Value
Alias Name	PHONENUMBER
SQL	APPS.YYY.GETCUSTOMERPHONE
Relation	CUSTOMER.ID

Child Alias 2

This child alias issues a SQL SELECT statement to retrieve address information for each customer that was returned in the parent alias's query. This alias can return more than one address, each of which will appear as documents within an ADDRESS document list in the results document.

Field	Value
Alias Name	ADDRESS
SQL	SELECT STREET FROM TABLE3
Relation	ID = CUSTOMER.ID

Note that the relation field refers to the ID field in the parent alias name, not to the database table name. At runtime the service will issue a SELECT statement as follows:

```
SELECT STREET FROM TABLE3 WHERE ID = valueFromParent
```

Results

Using the values supplied in the sample tables, the outcome of this transaction would look like this:

Name		Value
results		
CUSTOMER		
CUSTOMER[0]		
NAME		John Smith
ID		001
PHONE		(888) 555-1212
ADDRESS		
ADDRESS[0]		
STREET		Main Street
ADDRESS[1]		
STREET		Broadway Blvd.


Adapter Services

■ Overview	62
■ Before Configuring or Managing Adapter Services	62
■ Using Oracle Applications Adapter Services	63
■ Configuring Insert SQL Services	64
■ Configuring Compound Select SQL Services	65
■ Configuring Simple SQL Services	66
■ Testing Adapter Services	68
■ Viewing Adapter Services	69
■ Editing Adapter Services	69
■ Deleting Adapter Services	70
■ Enabling Automatic Data Validation	70
■ Reloading Adapter Values in Developer	71

Overview

This chapter describes how to configure and manage Oracle Applications Adapter services. For detailed descriptions of the available Oracle Applications Adapter services, see [“Adapter Services” on page 18](#).

Before Configuring or Managing Adapter Services

 To prepare to configure or manage Oracle Applications Adapter services

- 1 Start the Integration Server and the Integration Server Administrator, if they are not already running.
- 2 Make sure that you have Integration Server Administrator or Developer privileges so that you can access the Oracle Applications Adapter administrative screens. See the *webMethods Integration Server Administrator's Guide* for information about setting user privileges.
- 3 Using the Administrator, make sure the WmOAAAdapter package is enabled. See [“Enabling and Disabling Packages” on page 26](#) for instructions.
- 4 If you are using Integration Server version 6.1, make sure you are viewing the webMethods Developer in the Edit perspective, as described in [“Viewing Different Perspectives of the webMethods Developer” on page 22](#).
- 5 Using the Administrator, configure an adapter connection to use with the adapter service. See [“Configuring Adapter Connections” on page 35](#) for instructions.
- 6 Start the Developer if it is not already running.
- 7 Make sure that you enable the Automatic data validation. Refer to [“Enabling Automatic Data Validation” on page 70](#) for detailed instructions on how to enable automatic data validation.
- 8 Using the Developer, create a user-defined package to contain the service, if you have not already done so. When you configure adapter services, you should always define them in user-defined packages rather than in the WmOAAAdapter package.
- 9 Using the Developer, create a user-defined package to contain the connection, if you have not already done so. For more information about managing packages for the adapter, see [“Oracle Applications Adapter Package Management” on page 24](#).



Important! If the transaction definitions do not accurately reflect the Oracle Applications use of the DBMS, unpredictable results can occur when the service is run. The webMethods administrator should review the transaction definition with an Oracle Applications administrator to make sure they are suitable for each Oracle Applications Adapter service. For information about updating the transaction definitions, refer to [“Updating IS-to-Oracle Applications Transaction Definitions” on page 34](#) and [“Updating Oracle Applications-to-IS Transaction Definitions” on page 35](#).

Using Oracle Applications Adapter Services

The following table lists the tasks required to use adapter services:

Task	Use this tool...
1 Configure an adapter connection. See “Configuring Adapter Connections” on page 35 for details.	Integration Server Administrator (Adapters menu)
2 Import the transaction definitions or configure new transaction definitions to be used by the service. For instructions, see “Importing Transaction Definitions” on page 46 .	Integration Server Administrator (Adapters menu)
3 Select the appropriate adapter service template and configure the adapter service. Refer to the appropriate service section in this chapter for more information on configuring the specific adapter services.	Developer
4 If you plan to use an Integration Server flow or Java service to invoke the adapter service, design the flow or Java service to use this adapter service. See the <i>webMethods Developer User’s Guide</i> for details.	Developer
5 Manage the adapter service. See Chapter 2, “Package Management” on page 23 and Chapter 7, “Logging and Exception Handling” on page 83 for details.	Developer and Integration Server Administrator

Configuring Insert SQL Services

Insert services enable the adapter to insert records into the Oracle Applications open interface tables. These records will either insert new information or update existing information in the Oracle Applications system.

For more information about Insert SQL services, see [“Insert SQL” on page 19](#).

 To configure an Insert SQL service

- 1 From the Developer's File menu, select **New**.
- 2 Select **Adapter Service** from the list of elements, and then click **Next**.
- 3 Select **Oracle Apps Adapter** as the adapter type, and then click **Next**.
- 4 Select the appropriate **Adapter Connection Name**, and then click **Next**.
- 5 Select the **Insert SQL** template, and then click **Next**.
- 6 Select a package and folder to contain the service, type a unique name for the service, and then click **Finish**.

Select a package that contains only services that execute Oracle Applications transactions. By doing this, it is easier to backup and replicate those services because they are all contained in a separate package.



Important! Do not store the services in the Oracle Applications package (WmOAAdapter) because the package is replaced when you upgrade the Oracle Applications Adapter.

The Developer creates the service, and then the adapter service editor for the Insert SQL service appears.

- 7 Select the **Insert SQL** tab, if it is not already selected.
- 8 Select a transaction definition from the **Transaction** list.
- 9 You can select the **Adapter Settings** tab at any time to confirm adapter service properties such as the **Adapter Name**, **Adapter Connection Name**, and **Adapter Service Template**, as necessary.
- 10 For information about configuring the **Input/Output** tab, see the *webMethods Developer User's Guide*. This tab applies to all services that you configure using the Developer.


- 11 From the File menu, select **Save** (or **Save All**).

To test the service directly from the Developer, see [“Testing Adapter Services” on page 68](#).

You may want to also configure an Error service to verify that the information was correctly inserted. For more information, see [“Configuring Simple SQL Services” on page 66](#).

Configuring Compound Select SQL Services

Includes Select services. For more information about Compound Select SQL services, see [“Compound Select SQL” on page 19](#).

 To configure a Compound Select SQL service

- 1 From the Developer’s File menu, select **New**.
- 2 Select **Adapter Service** from the list of elements, and then click **Next**.
- 3 Select **Oracle Apps Adapter** as the adapter type, and then click **Next**.
- 4 Select the appropriate **Adapter Connection Name**, and then click **Next**.
- 5 Select the **Compound Select SQL** template, and then click **Next**.
- 6 Select a package and folder to contain the service, type a unique name for the service, and then click **Finish**.

Select a package that contains only services that execute Oracle Applications transactions. By doing this, it is easier to backup and replicate those services because they are all contained in a separate package.



Important! Do not store the services in the Oracle Applications package (WmOAAAdapter) because the package is replaced when you upgrade the Oracle Applications Adapter.

The Developer creates the service, and then the adapter service editor for the Compound Select SQL service appears.

- 7 Select the **Compound Select SQL** tab, if it is not already selected.
- 8 Select a transaction definition from the **Transaction** list.
- 9 You can select the **Adapter Settings** tab at any time to confirm adapter service properties such as the **Adapter Name**, **Adapter Connection Name**, and **Adapter Service Template**, as necessary.

10 For information about configuring the **Input/Output** tab, see the *webMethods Developer User's Guide*. This tab applies to all services that you configure using the Developer.

11 From the File menu, select **Save** (or **Save All**).

To test the service directly from the Developer, see [“Testing Adapter Services” on page 68](#).

Configuring Simple SQL Services

Use Simple SQL services to create Acknowledge services and Error services.

For more information about Simple SQL services, see [“Simple SQL” on page 19](#).

 To configure an Acknowledge service

- 1 From the Developer's File menu, select **New**.
- 2 Select **Adapter Service** from the list of elements, and then click **Next**.
- 3 Select **Oracle Apps Adapter** as the adapter type, and then click **Next**.
- 4 Select the appropriate **Adapter Connection Name**, and then click **Next**.
- 5 Select the **Simple SQL** template, and then click **Next**.
- 6 Select a package and folder to contain the service, type a unique name for the service, and then click **Finish**.

Select a package that contains only services that execute Oracle Applications transactions. By doing this, it is easier to backup and replicate those services because they are all contained in a separate package.



Important! Do not store the services in the Oracle Applications package (WmOAAAdapter) because the package is replaced when you upgrade the Oracle Applications Adapter.

The Developer creates the service, and then the adapter service editor for the Simple SQL service appears.

- 7 Select the **Simple SQL** tab, if it is not already selected.
- 8 Select **AcknowledgeSQL** from the **Simple SQL Type** list.
- 9 Select a transaction definition from the **Transaction** list.



Note: When calling a stored procedure, ignore the output.

- 10 You can select the **Adapter Settings** tab at any time to confirm adapter service properties such as the **Adapter Name**, **Adapter Connection Name**, and **Adapter Service Template**, as necessary.
- 11 For information about configuring the **Input/Output** tab, see the *webMethods Developer User's Guide*. This tab applies to all services that you configure using the Developer.
- 12 From the **File** menu, select **Save** (or **Save All**).

To test the service directly from the Developer, see [“Testing Adapter Services” on page 68](#).

To configure an Error service

- 1 From the Developer's **File** menu, select **New**.
- 2 Select **Adapter Service** from the list of elements, and then click **Next**.
- 3 Select **Oracle Apps Adapter** as the adapter type, and then click **Next**.
- 4 Select the appropriate **Adapter Connection Name**, and then click **Next**.
- 5 Select the **Simple SQL** template, and then click **Next**.
- 6 Select a package and folder to contain the service, type a unique name for the service, and then click **Finish**.

Select a package that contains only services that execute Oracle Applications transactions. By doing this, it is easier to backup and replicate those services because they are all contained in a separate package.



Important! Do not store the services in the Oracle Applications package (WmOAAAdapter) because the package is replaced when you upgrade the Oracle Applications Adapter.

The Developer creates the service, and then the adapter service editor for the Simple SQL service appears.

- 7 Select the **Simple SQL** tab, if it is not already selected.
- 8 Select **ErrorSQL** from the **Simple SQL Type** list.
- 9 Select a transaction definition from the **Transaction** list.
- 10 You can select the **Adapter Settings** tab at any time to confirm adapter service properties such as the **Adapter Name**, **Adapter Connection Name**, and **Adapter Service Template**, as necessary.
- 11 For information about configuring the **Input/Output** tab, see the *webMethods Developer User's Guide*. This tab applies to all services that you configure using the Developer.

12 From the File menu, select **Save** (or **Save All**).

To test the service directly from the Developer, see [“Testing Adapter Services” on page 68](#).


Testing Adapter Services

You use the Developer to test adapter services. If you are using Integration Server version 6.1, make sure you are viewing the webMethods Developer in the Edit perspective, as described in [“Viewing Different Perspectives of the webMethods Developer” on page 22](#).

To test adapter services in the Developer's Adapter Service Editor

- 1 In the Developer's Navigation Panel, expand the package and folder that contain the service you want to test.
- 2 Double-click the service you want to test.

The Developer displays the configured service in the service template's Adapter Service Editor.

- 3 Click the **Run** icon .
- 4 For every service input field, you will be prompted to enter an input value. Enter a value for each input field and then click **OK**.
- 5 Click the **Results** tab to view the output from this service.

For more information about testing and debugging services, see the *webMethods Developer User's Guide*.

Viewing Adapter Services

You use the Developer to view adapter services. If you are using Integration Server version 6.1, make sure you are viewing the webMethods Developer in the Edit perspective, as described in [“Viewing Different Perspectives of the webMethods Developer” on page 22](#).

To view a service

- 1 In the Administrator, enable the connection for the service you want to edit. Refer to [“Enabling Adapter Connections” on page 43](#) for instructions on enabling connections.
- 2 In the Developer’s Navigation Panel, expand the package and folder that contain the service you want to view.
- 3 Double-click the service you want to view.

The Developer displays the configured service in the service template’s Adapter Service Editor.

Editing Adapter Services

You use the Developer to edit adapter services. If you are using Integration Server version 6.1, make sure you are viewing the webMethods Developer in the Edit perspective, as described in [“Viewing Different Perspectives of the webMethods Developer” on page 22](#).

Depending on which version of the Integration Server you use, you may be able to change the connection associated with an adapter service, as follows:

- When using Integration Server 6.0.1 or earlier, you *cannot* change which connection an adapter service uses after the service is configured.
- When using Integration Server version 6.1, you *can* change which connection an adapter service uses. To do this, you use the built-in service `setAdapterServiceNodeConnection`. For more information, see [“Changing the Connection Associated With an Adapter Service or Notification at Design Time” on page 16](#).

To edit a service

- 1 In the Administrator, enable the connection for the service you want to edit. Refer to [“Enabling Adapter Connections” on page 43](#) for instructions on enabling connections.
- 2 In the Developer’s Navigation Panel, expand the package and folder that contain the service that you want to edit.

- 3 Double-click the service you want to edit.

The Developer displays the configured service in the service template's Adapter Service Editor.

- 4 Lock the selected service for editing by right-clicking on the selected service and then selecting **Lock for Edit**.
- 5 Modify the values for the service's parameters as needed. For detailed descriptions of the service's parameters, see the section on configuring a service for the specific type of service you want to edit.
- 6 Save the service.
- 7 Unlock the service by right-clicking on the service and then selecting **Unlock**.

Deleting Adapter Services

You use the Developer to delete adapter services. If you are using Integration Server version 6.1, make sure you are viewing the webMethods Developer in the Edit perspective, as described in [“Viewing Different Perspectives of the webMethods Developer” on page 22](#).



To delete a service

- 1 In the Developer's Navigation Panel, expand the package and folder that contain the service you want to delete.
- 2 Right-click the service you want to delete and then select **Delete**.

Enabling Automatic Data Validation

You can configure the Developer to validate user-defined data in adapter services at design time. Oracle Applications Adapter requires that you enable the **Automatic Data Validation** option before you create your adapter services.

To validate all values set for the adapter service, select from the Developer the **Tools > Options > Integration Server > Adapter Service/Notification Editor** item and enable the **Automatic data validation** option.



Note: When you select the option to always validate values for all adapter services, it will do so for all webMethods 6.x adapters installed on the Integration Server, which could potentially slow design-time operations for adapters.

Reloading Adapter Values in Developer

The Developer enables the Oracle Applications Adapter to reload and validate user-defined data for adapter services at design time. You can reload values for a single adapter service or you can configure the Developer so it automatically reloads the values for all adapter services.



Note: If you select the option to automatically reload values for adapter services, it will do so for all webMethods 6.x adapters installed on the Integration Server, which could potentially slow your design-time operations.

To reload the adapter values and re-validate all user values as needed for the adapter service, select from the Developer the **Tools > Options > Integration Server > Adapter Service/Notification Editor** item and enable the **Automatic polling of adapter metadata** option.

When using the Integration Server version 6.0.1 and earlier, this option is also available as the **Refresh** icon on the Developer.

This option enables data validation for the selected adapter service only. It compares the service values against the resource data that has already been fetched from the adapter.

See the *webMethods Developer User's Guide* for more information about the **Adapter Service/Notification Editor** and other Developer menu options and toolbar icons.


Adapter Notifications

■ Overview	74
■ Before Configuring or Managing Notifications	74
■ Configuring Notifications	75
■ Managing Notifications	76
■ Exporting Configured Adapter Notifications	78
■ Viewing Notifications	79
■ Editing Notifications	79
■ Deleting Notifications	80
■ Validating Adapter Notification Values	80
■ Reloading Adapter Values	81

Overview

This chapter describes how to configure and manage Oracle Applications Adapter notifications. For detailed descriptions of the available Oracle Applications Adapter notifications, see [“Adapter Notifications” on page 20](#).

Before Configuring or Managing Notifications

 To prepare to configure or manage Oracle Applications Adapter notifications

- 1 Start your Integration Server and the Administrator, if they are not already running.
- 2 Make sure you have Integration Server Administrator and Developer privileges so that you can access the Oracle Applications Adapter’s administrative screens. See the *webMethods Integration Server Administrator’s Guide* for information about setting user privileges.
- 3 Using the Administrator, make sure the WmOAAadapter package is enabled. See [“Enabling and Disabling Packages” on page 26](#) for instructions.
- 4 Using the Administrator, configure an adapter connection to use with the notification. See [“Configuring Adapter Connections” on page 35](#) for instructions.



Note: When using versions of the Integration Server earlier than 6.1, you may *not* assign a different connection to a polling notification after you configure the notification. You may, however, change the parameters of the connection.

To solve this limitation, Integration Server 6.1 provides a built-in service that you can use at design time to change the connection associated with a polling notification. For more information, see [“Changing the Connection Associated With an Adapter Service or Notification at Design Time” on page 16](#).

- 5 Start the Developer if it is not already running.
- 6 If you are using the Developer 6.1, make sure you are viewing the webMethods Developer in the Edit perspective, as described in [“Viewing Different Perspectives of the webMethods Developer” on page 22](#).
- 7 Using the Developer, create a user-defined package to contain the notification, if you have not already done so. When you configure notifications, you should always define them in user-defined packages rather than in the WmOAAadapter package. For more information about managing packages for the adapter, see [Chapter 2, “Package Management” on page 23](#).
- 8 You must schedule a notification and then enable it before you can use the notification. See [“Managing Notifications” on page 76](#) for instructions.

Configuring Notifications

Be sure to review the section [“Before Configuring or Managing Notifications” on page 74](#) before you configure notifications.

To configure a notification

- 1 From the Developer’s File menu, select **New**.
- 2 Select **Adapter Notification** from the list of elements and click **Next**.
- 3 Select **Oracle Apps Adapter** as the adapter type and click **Next**.
- 4 Select **Polling Notification** as the template and click **Next**.
- 5 Select the appropriate **Adapter Connection Name** and click **Next**.
- 6 Select a package and folder to contain the notification, type a unique name for the notification, and then click **Next**.

Select a package that contains only notifications that execute Oracle Applications transactions. By doing this, it is easier to backup and replicate those notifications because they are all contained in a separate package.



Important! Do not store the notifications in the Oracle Applications package (WmOAAAdapter) because the package is replaced when you upgrade the Oracle Applications Adapter.

- 7 The name of the publishable document associated with this notification displays. Click **Finish**.

For more information about adapter notifications and publishable documents, see [“Adapter Notifications” on page 20](#). For more details about Integration Server publishable documents when using versions of the Integration Server earlier than 6.1, see the *Building Integration Solutions Using Publication* document, and when using Integration Server 6.1, see the *Publish-Subscribe Developer’s Guide*.

- 8 The Developer creates the notification, and the editor for the adapter notification appears.
 - You can select the **Adapter Settings** tab at any time to confirm adapter notification properties such as the **Adapter Name**, **Adapter Connection Name**, and **Adapter Notification Template**, as necessary.
 - For information about using the **Permissions** tab to assign an access control list (ACL) to an element using versions of the Developer earlier than 6.1, see the *webMethods Developer User’s Guide*.

When using Developer 6.1, the information from the **Permissions** tab now appears in the **Properties** panel.

- 9 Select the **Polling Notification** tab, if it is not already selected.
- 10 Select a transaction definition from the **Transaction** list. The editor completes the remaining fields based on the transaction selected.
- 11 From the **File** menu, select **Save** (or **Save All Editors**).
- 12 You must create a trigger for the publishable document associated with the notification. For more information about creating triggers and working with publishable documents, see the *Publish-Subscribe Developer's Guide*.
- 13 By default a notification is disabled when it is configured. You must schedule and enable the notification using the Integration Server Administrator before you can use it. See [“Managing Notifications” on page 76](#) for details.

Managing Notifications

Using the Integration Server Administrator you can display all polling notifications that have been configured for the Oracle Applications Adapter. You can also enable and disable a notification and schedule it for execution.

You must schedule a notification and then enable it before you can use the notification.






Note: You must have webMethods administrator privileges to access the Oracle Applications Adapter's administrative screens. See the *webMethods Integration Server Administrator's Guide* for information about setting user privileges.




To manage polling notifications

- 1 Start the Integration Server Administrator.
- 2 From the **Adapters** menu in the navigation area of the Administrator, select **Oracle Apps Adapter**.
- 3 From the navigation area, select **Polling Notifications**. All configured Oracle Applications Adapter notifications are displayed.

- 4 From the Oracle Apps Adapter Polling Notifications table, use the fields in the following table to manage each adapter notification:

Field	Description/Action
Notification Name	The name of the notification.
Package Name	The name of the package for the notification.
Enabled	<p>After you schedule a polling notification, you can use this option to enable (Yes) or disable (No) a polling notification. Click on the current value in this field to change its value.</p> <p>If there is no polling notification scheduled for a given adapter notification, Not Scheduled appears in this field. Use the Edit Schedule  icon to specify a schedule as described in step 5.</p> <hr/> <p>Note: You must schedule a polling notification before you can enable it.</p>
Edit Schedule	<p>Click the Edit Schedule  icon to create or modify polling notification parameters. Use the instructions in step 5 to create or modify a notification's schedule parameters.</p> <hr/> <p>Note: You must disable a polling notification before you can edit it.</p>
View Schedule	<p>Click the View Schedule  icon to review the parameters for the selected polling notification. If the notification has not been scheduled, the icon is greyed out.</p> <p>Click Return to Oracle Apps Adapter Notifications to go back to the main polling notification page.</p>

- 5 To create or modify schedule parameters for the selected adapter notification:
 - a Click the **Edit Schedule**  icon and specify the following schedule parameters:

Field	Description/Action
Interval (seconds)	Enter the polling interval time in seconds. By default the next polling interval starts after the current execution is completed.
Overlap	Note: Do not use this option; otherwise, when you enable this notification, it may lock up tables and cause the Integration Server to fail.
Immediate	Enable this option to start polling immediately. By default the polling interval begins when the notification is enabled but enabling this option causes the notification to start immediately after it is enabled.

- b When you are finished, click **Save Schedule**.
- 6 After you configure or modify a polling notification, you can enable it. Use the **Enabled** field described in [step 4](#) to enable a polling notification.

Exporting Configured Adapter Notifications

You can export notifications from one Integration Server to another Integration Server. You do not need to disable notifications in order to export them. In most cases, the current state of the notifications in the package that you export is retained. However, if you deploy to a different Integration Server and connect to a different database, then you should first disable the notification.



Note: A given notification can only run on one Integration Server at a time.

For more information about exporting packages, see the *webMethods Integration Server Administrator's Guide*.

Viewing Notifications

You use the Developer to view notifications. If you are using the Developer 6.1, make sure you are viewing the webMethods Developer in the Edit perspective, as described in [“Viewing Different Perspectives of the webMethods Developer” on page 22](#).

To view a notification

- 1 In the Administrator, enable the connection for the notification you want to edit. Refer to [“Enabling Adapter Connections” on page 43](#) for instructions on enabling connections.
- 2 In the Developer’s Navigation Panel, expand the package and folder that contain the notification you want to view.
- 3 Select the notification you want to view.

The Developer displays the notification in the notification template’s Adapter Notification Editor.

Editing Notifications

You use the Developer to edit notifications. If you are using the Developer 6.1, make sure you are viewing the webMethods Developer in the Edit perspective, as described in [“Viewing Different Perspectives of the webMethods Developer” on page 22](#).

Depending on which version of the Integration Server you use, you may be able to change the connection associated with an adapter notification, as follows:

When using versions of the Integration Server earlier than 6.1, you *cannot* change which connection an adapter notification uses after the notification is configured.

- When using Integration Server 6.1, you *can* change which connection an adapter notification uses. To do this, you use the built-in service `setAdapterServiceNodeConnection`. For more information, see [“Changing the Connection Associated With an Adapter Service or Notification at Design Time” on page 16](#).

To edit a notification

- 1 In the Administrator, enable the connection for the notification you want to edit. Refer to [“Enabling Adapter Connections” on page 43](#) for instructions on enabling connections.
- 2 In the Developer’s Navigation Panel, expand the package and folder that contain the notification you want to view.

- 3 Select the notification you want to edit.

The Developer displays the notification in the notification template's Adapter Notification Editor.

- 4 Lock the selected notification for editing by right-clicking on the selected notification and then selecting **Lock for Edit**.
- 5 Modify the values for notification's parameters as needed. For detailed descriptions of the notification's parameters, see the section on configuring a notification for the specific type of notification you want to edit.

Deleting Notifications

You use the Developer to delete adapter notifications. If you are using the Developer 6.1, make sure you are viewing the Developer in the Edit perspective, as described in [“Viewing Different Perspectives of the webMethods Developer” on page 22](#).

Deleting a notification also deletes the associated publishable document.

To delete a notification

- 1 In the Administrator, disable the notification. Otherwise, the trigger and buffer table created by the notification will remain in the database. To disable a notification, see [“Managing Notifications” on page 76](#).
- 2 In the Developer's Service Browser, expand the package and folder that contain the notification you want to delete.
- 3 Right-click the notification and then click **Delete**.


Validating Adapter Notification Values

The Developer enables the Oracle Applications Adapter to validate user-defined data for adapter notifications at design time. You can validate the values for a single notification or you can configure the Developer to always validate the values for notifications.



Note: If you select the option to always validate values for adapter notifications, it will do so for all webMethods 6.x adapters installed on the Integration Server, which could potentially slow your design-time operations.

To validate all values, select from the Developer the **Tools > Options > Integration Server > Adapter Service/Notification Editor** item and enable the **Automatic data validation** option.

When using versions of the Integration Server earlier than 6.1, this option is also available as an icon on the Developer, the **Validate service/notification value** icon . This icon is available only when the **Automatic data validation** option is currently disabled.

The **Automatic data validation** option enables data validation for the selected adapter notification only. It compares the service values against the resource data that has already been fetched from the adapter. Note that this option can slow operations.

See the *webMethods Developer User's Guide* for more information about the **Adapter Service/Notification Editor** and other Developer menu options and toolbar icons.


Reloading Adapter Values

The Developer enables the Oracle Applications Adapter to reload and validate user-defined data for notifications at design time. You can reload values for a single notification or you can configure the Developer so it automatically reloads the values for adapter notifications.



Note: If you select the option to automatically reload values for notifications, it will do so for all webMethods 6.x adapters installed on the Integration Server, which could potentially slow your design-time operations.

To reload the adapter values and revalidate all user values as needed for the adapter service, select from the Developer the **Tools > Options > Integration Server > Adapter Service/Notification Editor** item and enable the **Automatic polling of adapter metadata** option.

When using versions of the Integration Server earlier than 6.1, this option is also available as an icon on the Developer, the **Refresh** icon .

This option enables data validation for the selected adapter service only. It compares the service values against the resource data that has already been fetched from the adapter.

See the *webMethods Developer User's Guide* for more information about the **Adapter Service/Notification Editor** and other Developer menu options and toolbar icons.

Logging and Exception Handling

■ Overview	84
■ Oracle Applications Adapter Message Logging	84
■ Handling Oracle Applications Errors	85
■ Oracle Applications Adapter Error Codes	86

Overview

The following sections describe message logging and exception handling for the Oracle Applications Adapter. A list of error codes and supporting information appears in [“Oracle Applications Adapter Error Codes” on page 86](#).

Oracle Applications Adapter Message Logging

The Oracle Applications Adapter uses the Integration Server’s logging mechanism to log messages. You can configure and view the Integration Server’s logs to monitor and troubleshoot the Oracle Applications Adapter. See the *webMethods Integration Server Administrator’s Guide* for detailed information about logging in the Integration Server, including instructions for configuring and viewing the different kinds of logs supported by the server.

The Integration Server maintains several types of logs; however, the Oracle Applications Adapter only logs messages to the Audit, Error and Server logs, as described in the table below:

Log	Description
Audit Log	You can monitor individual adapter services using the audit log as you would audit any service in the Integration Server. The audit properties for an adapter service are available in each Oracle Applications Adapter service template on the Audit tab.
Error Log	The Oracle Applications Adapter automatically posts critical-level and error-level log messages to the server's Error log. These log messages will appear as Adapter Runtime messages.
Server Log	The Oracle Applications Adapter posts messages to the Server log, depending on how the server log is configured. Critical-level through debug-level log messages appear as Adapter Runtime messages. V1-Verbose1 or V4-Verbose4 log messages appear as Oracle Applications Adapter log messages.

The Oracle Applications Adapter's log messages appear in the following format: ADA.0336.nnnmc, where:

- ADA is the facility code that indicates that the message is from an adapter.
- 0336 (or 336) is the Oracle Applications Adapter major code, which indicates that the error is generated by the Oracle Applications Adapter.

- *nnnn* represents the error's minor code. For detailed descriptions of the Oracle Applications Adapter's minor codes, see [“Oracle Applications Adapter Error Codes” on page 86](#).
- *c* represents the message's severity level (optional).

Because the Oracle Applications Adapter works in conjunction with the WmART package, the adapter's messages and exceptions typically appear within log messages for the WmART package, although that is not necessarily true for verbose-level messages.

To monitor the Oracle Applications Adapter's log messages in the Server log, ensure that your server log's logging settings are configured to monitor the following facilities:

- 0113 Adapter Runtime (Managed Object)
- 0114 Adapter Runtime
- 0115 Adapter Runtime (Listener)
- 0116 Adapter Runtime (Notification)
- 0117 Adapter Runtime (Adapter Service)
- 0118 Adapter Runtime (Connection)
- 0121 Adapter Runtime (SCC Transaction Manager)
- 0126 Adapter Runtime (SCC Connection Manager)

Handling Oracle Applications Errors

If an Oracle Applications Adapter object (for example, a connection or service) encounters an error with the Oracle Applications system, the object will throw an adapter error coupled with the Oracle Applications error, exactly as it was thrown by Oracle Applications. The Oracle Applications errors will be in the IData format.

For example, if an adapter service fails on the Oracle Applications system at run time because its input fields contain invalid values, you will receive an adapter error that indicates that the service failed, and the adapter error will contain the specific error generated on the Oracle Applications system indicating why the service failed. In this case, you would receive an Oracle Applications error specifying that the input values are invalid.

Oracle Applications Adapter Error Codes

This section lists the Oracle Applications Adapter's Critical- and Error-level minor codes, including the text of each error, information about the cause of the errors, and possible responses to the errors.

0 Missing required input: *value*.

Cause: The specified required input is missing.

Response: Provide the required input and then retry the operation.

3 *OARelease* is an unsupported Oracle Applications release.

Cause: The adapter does not support your version of Oracle Applications.

Response: See the 6.0 *webMethods Oracle Applications Adapter Installation Guide* for the supported versions of Oracle Applications.

4 Unable to connect to the repository to perform the *operation*. Error logged.

Cause: While importing or exporting transaction definition files, the adapter is not able to connect to the repository server.

Response: Check the remote repository server and ensure that it is running. Check the network and ensure that there are no network problems.

5 Unable to obtain a lock on the repository to perform the *operation*. Error logged.

Cause: While importing or exporting transaction definition files, the adapter is not able to lock the repository to update the transaction definition. This error may be caused when multiple users are importing or exporting large transaction definition files at the same time.

Response: Determine whether another user is performing the same operation.

6 Error accessing the repository. Error logged.

Cause: The adapter is not able to operate on the remote repository server.

Response: Check remote repository server and ensure that it is running. Check the network and ensure that there are no network problems.

7 *transactionDefinitionFile* already exists.

Cause: The indicated transaction definition file (.txp) already exists.

Response: Check the file names and change names that are the same.

8 *transactionName* does not exist.

Cause: The indicated required configuration information is not available. This error may be caused by an incorrectly configured service or notification.

Response: Check the service or notification configuration and verify that the information is correct.

9 Unable to retrieve version - error logged.

Cause: The adapter cannot retrieve the WmOAAadapter version number.

Response: Replace the corrupted manifest file.

10 Unable to retrieve build - error logged.

Cause: The adapter cannot retrieve the WmOAAadapter build number.

Response: Replace the corrupted manifest file.

100 Error rolling back tran: *transaction*.

Cause: The adapter encountered an error while trying to roll back the indicated transaction. This error may be caused by the adapter connection being lost.

Response: Check the database log to see the error information, and then address the error as necessary.

101 No connection found.

Cause: An invalid connection was specified or the connection is no longer available.

Response: Specify a valid connection.

102 Number of data values does not match the number of variables.

Cause: The number of data values supplied to the service does not match the required number of variables.

Response: Ensure that the number of data values matches the number of variables.

103 Method only valid on raw SQL commands.

Cause: The Oracle Applications-to-IS transaction definition incorrectly uses SQL statements and stored procedures.

Response: You cannot combine stored procedures with SQL queries in an Oracle Applications-to-IS transaction definition. Use only stored procedures or SQL trees. See [“Creating Oracle Applications-to-IS Transaction Definitions” on page 51](#) for more information.

104 Method only valid on Stored Procedure commands.

Cause: The Oracle Applications-to-IS transaction definition incorrectly uses SQL statements and stored procedures.

Response: You cannot combine stored procedures with SQL queries in an Oracle Applications-to-IS transaction definition. Use only stored procedures or SQL trees. See [“Creating Oracle Applications-to-IS Transaction Definitions” on page 51](#) for more information.

110 Exception caught: *exceptionMessage*.

Cause: A Java Exception has been caught.

Response: Determine the cause of the exception and correct the problem. If necessary, provide this information to a webMethods Technical Services representative.

111 SQL Exception caught: *exceptionMessage*.

Cause: A SQL statement execution exception has been caught.

Response: Determine the cause of the exception and correct the problem. Refer to the Oracle Database Manual for any error code specified in the *exceptionMessage*. If necessary, provide this information to a webMethods Technical Services representative.

112 An inbound SQL statement must start with INSERT, UPDATE, DELETE or be a stored procedure call.

Cause: An IS-to-Oracle Applications transaction definition contains an invalid SQL statement. The invalid SQL statement does not start with INSERT, UPDATE, DELETE, or a stored procedure call.

Response: Correct the statement in the transaction definition by starting it with INSERT, UPDATE, DELETE, or a stored procedure call.

113 An outbound SQL statement must start with SELECT or be a stored procedure call.

Cause: An Oracle Applications-to-IS transaction definition contains an invalid SQL statement. The invalid SQL statement does not start with SELECT or is not a stored procedure call.

Response: Examine the transaction definition, and correct the SQL statement or use a stored procedure call.

114 Ill-formed stored procedure call *storedProcedure*.

Cause: The statement to refer to the Stored Procedure is syntactically invalid.

Response: Correct the statement in the transaction definition.

118 *variableText* missing from transaction definition.

Cause: The indicated information is missing from the transaction definition.

Response: Add the missing information to the transaction definition.

214 Error executing SQL for alias *nodeName*, *SQLErrorMessage*.

Cause: Configure time error. The adapter is not able to execute the SQL statement defined under an alias node of the SQL tree of an Oracle Applications-to-IS transaction definition.

Response: Correct the transaction definition.

300 Error saving export file: *filename* - error logged.

Cause: The adapter cannot create the export file correctly.

Response: Possibly a file system access privilege issue, or not enough disk space.

301 File name *importTransactionDefFile* contains invalid characters.

Cause: The file contains invalid characters.

Response: The characters are invalid for Oracle or invalid for the adapter.

302 Invalid file contents. Missing *tagName* tag.

Cause: While importing a transaction definition transport file (.txp), the adapter detected invalid contents in the file.

Response: Correct the transaction definition transport file and reimport it, or remove the file from the *IntegrationServer_Directory*\packages\WmOAAAdapter\exchange directory.

303 Invalid file contents. Missing *valueName* value.

Cause: While importing a transaction definition transport file (.txp), the adapter detected invalid contents in the file.

Response: Correct the transaction definition transport file and reimport it, or remove the file from the *IntegrationServer_Directory*\packages\WmOAAAdapter\exchange directory.

304 Unsupported Oracle Applications Release *version* found in file.

Cause: While importing a transaction definition transport file (.txp), the adapter found a version of Oracle Applications that it does not support.

Response: Examine the transaction definition file and specify a supported version of Oracle Applications. See the *webMethods Oracle Applications Adapter Installation Guide* for the supported versions.

305 *fileName* file not found.

Cause: The adapter did not find this file.

Response: Add this file to the appropriate directory.

306 Error reading *fileName*. *IOErrorMessage*.

Cause: Generic message format.

Response: Take the appropriate action based on the *IOErrorMessage*.

307 Invalid transaction definition: *transactionName* is missing the *tagName* tag.

Cause: The transaction definition file is missing the indicated tag.

Response: Add the indicated tag to the transaction definition.

308 Invalid transaction definition: *transactionName* has an unknown type of *transactionType*.

Cause: The transaction definition is of an unknown type.

Response: Assign a valid type (Oracle-to-IS or IS-to-Oracle) to the transaction definition.

310 The load() method must be called before this method.

Cause: The transaction definition is not loaded in the repository

Response: Import the transaction definition into the repository.

311 *errSQL* is only valid on Oracle-to-IS transactions.

Cause: The transaction definition file has an invalid *errSQL* tag. This error is typically caused by manually editing the transaction definition file and specifying an incorrect structure.

Response: Locate the original transaction definition file and use it instead of the edited file.

312 *insertSQL* is only valid on IS-to-Oracle transactions.

Cause: The transaction definition file has an invalid *insertSQL* tag. This error is typically caused by manually editing the transaction definition file and specifying an incorrect structure.

Response: Locate the original transaction definition file and use it instead of the edited file.

313 Error in the SQLPath. Unable to find alias *aliasName* at the following level *aliasName*.

Cause: This is an Internal Error, and indicates a corrupted service configuration. This error could be caused by the contents of the repository being changed after the service has been configured, resulting in corrupt service information.

Response: Refresh the service. If the error persists, delete the service and reconfigure it.

314 Transaction definition is missing the *tagName* tag.

Cause: The transaction definition is missing the indicated tag.

Response: Add the missing tag to the transaction definition.

316 Unable to update transaction definition. Missing transaction type.

Cause: The transaction definition file is missing the transaction type tag. This error is typically caused by manually editing the transaction definition file.

Response: Locate the original transaction definition file and use it instead of the edited file.

317 Missing transaction name.

Cause: The transaction definition file is missing the transaction name tag. This error is typically caused by manually editing the transaction definition file.

Response: Locate the original transaction definition file and use it instead of the edited file.

318 Missing SQL Alias.

Cause: The transaction definition file is missing the SQL Alias tag. This error is typically caused by manually editing the transaction definition file.

Response: Locate the original transaction definition file and use it instead of the edited file.

600 Error encountered in alias *aliasName*: *SQLErrorMessage*.

Cause: During design time, the adapter could not execute the SQL statement associated with the indicated alias. Typically this error occurs because the SQL statement is invalid.

Response: Examine the transaction definition and check the SQL statement associated with the alias. Correct it as necessary.

601 No data found for insertion.

Cause: The data for insertion is not provided.

Response: Examine the service invocation and provide the necessary data for insertion.

602 Illformed transactionRecord input.

Cause: The input for the transaction is not in the expected structure. This error typically indicates a data mapping problem.

Response: Examine the service invocation and provide the proper data mapping.

603 Data inputs for required tables are not available.

Cause: Run-time error. The required input data not provided for the INSERT. This error is typically indicates a data mapping problem.

Response: Examine the service invocation mapping and provide the required input data.

604 Error log sql execution failed - error logged.

Cause: The adapter service is not able to perform the SQL SELECT for the error message that is logged in the Oracle Applications concurrent process execution.

Response: Check the IS error log for further details.

605 Unable to perform insert - error logged.

Cause: The adapter is not able to perform the SQL INSERT operation.

Response: Check the IS error log for further details.

606 Unable to perform select - error logged.

Cause: The adapter is not able to perform the SQL SELECT operation.

Response: Check the IS error log for further details.

943 Attempting to get lock and update. Retry: {0}.

Cause: The adapter cannot lock the repository to update the transaction definition.

Response: Make sure no other users are accessing the repository for the transaction definition being updated.

961 Invalid token count.

Cause: Invalid Stored Procedure invocation syntax in the SQL statement of the Oracle-to-IS transaction definition.

Response: Check the transaction definition and correct the stored procedure syntax.

979 Illegal character string: *characterString*.

Cause: The transaction definition transport (.txp) file contents contain invalid characters.

Response: Examine the transaction definition file and specify a valid transport file name.

1000 Empty Error SQL.

Cause: Error SQL is not specified in the transaction definition.

Response: Check the transaction definition and include the necessary Error SQL.

1001 No parameter named *parameterName*.

Cause: Run time error. The adapter cannot find the input parameter required by the stored procedures in the service invocations.

Response: Check the service invocation mapping.

1002 Parameter *index* has no name.

Cause: Run time error. The adapter cannot retrieve the parameter name for a stored procedure from the Oracle database. Typically this error is caused by using an inappropriate JDBC driver.

Response: Use the correct JDBC driver. See the *webMethods Oracle Applications Adapter Installation Guide* for the correct JDBC driver.

1003 Resource Domain Lookup Failed *errorMessage*.

Cause: The adapter is not able to configure the service.

Response: Turn on debug for further details.

1008 Transaction Names are not Available.

Cause: Transaction definitions may have been deleted from the repository.

Response: Check the repository to see if the transaction definition is available.

1010 Select Transaction.

Cause: While configuring an adapter service, you did not select a transaction definition.

Response: Select a transaction.

1210 Transaction Names are not available.

Cause: The transaction definition does not exist in the repository.

Response: Check the transaction definition and specify the transaction names.

1211 Tables are not available.

Cause: For an IS-to-Oracle transaction definition, there is no table specified for insertion.

Response: Check the transaction definition and specify a table.

338 Unable to rollback.

Cause: This error indicates an Execute SQL rollback exception, and is possibly due to the adapter connection being lost.

Response: Check your network and database server.

8106 Failed to commit local transaction. *errorMessage*.

Cause: This error indicates an Execute SQL commit exception, and is possibly due to the adapter connection being lost.

Response: Check your network and database server.

8107 Failed to rollback local transaction. *errorMessage*.

Cause: This error indicates an Execute SQL rollback exception, and is possibly due to the adapter connection being lost.

Response: Check your network and database server.

8200 The JDBC DataSource class *className* cannot be located.

Cause: A DataSource class name was specified in the adapter Connection Properties DataSource Class field, but the class cannot be located. Either the class does not exist or the name was misspelled.

Response: Check the spelling and make sure the JDBC driver file is in the CLASSPATH or in the *IntegrationServer_directory/packages/WmOAAadapter/code/jars* directory.

8201 The JDBC DataSource class *className* cannot be instantiated.

Cause: The JDBC driver's DataSource class failed to initialize.

Response: Use the correct JDBC driver. See the *webMethods Oracle Applications Adapter Installation Guide* for the correct JDBC driver.

8203 Illegal property for JDBC DataSource class *propertyName*.

Cause: An invalid property for the Datasource class was specified for the connection.

Response: Refer to JDBC Driver manual for eligible properties.

8207 The JDBC DataSource class *className* does not support LOCAL_TRANSACTION.

Cause: This error is typically caused by using an inappropriate JDBC driver.

Response: Use the correct JDBC driver. See the *webMethods Oracle Applications Adapter Installation Guide* for the correct JDBC driver.

8208 Cannot disconnect from the database *errorMessage*.

Cause: A JDBC driver exception was thrown while the adapter was attempting to close the database connection. This error may be caused by a lost connection or the connection is already closed.

Response: Check your network and database server.

8312 Cannot commit the transaction to the database *errorMessage*.

Cause: This error indicates an Execute SQL commit exception. Possible reasons for the error are that the connection is lost or the Commit was inappropriately executed.

Response: Check your network and database server.

8319 Cannot get the list of table columns *errorMessage*.

Cause: While configuring an Insert SQL adapter service, the adapter cannot retrieve the table meta data. Typically this error is caused by using an inappropriate JDBC driver.

Response: Use the correct JDBC driver. See the *webMethods Oracle Applications Adapter Installation Guide* for the correct JDBC driver.

8320 Transaction File *transactionDefinitionFileName* has problems. Please correct it or remove this file.

Cause: While importing a transaction definition transport file (.txp), the adapter detected invalid contents in the file.

Response: Correct the transaction definition transport file and reimport it, or remove the file from the *IntegrationServer_Directory\packages\WmOAAAdapter\exchange* directory.

Built-In Transaction Management Services

- Transaction Management Overview 98
- Changing the Integration Server's Transaction Timeout Interval 100

Transaction Management Overview

This appendix provides an overview of using transactions. It describes how the Integration Server supports the built-in services in the WmART package used to manage explicit transactions for your Oracle Applications Adapter services. See the *webMethods Integration Server Built-In Services Reference* for descriptions of each of the specific built-in transaction management services.

Transactions

The Integration Server considers a transaction to be one or more interactions with one or more resources that are treated as a single logical unit of work. The interactions within a transaction are either all committed or all rolled back. For example, if a transaction includes multiple database inserts, and one or more inserts fail, all inserts are rolled back.

The Integration Server supports a *local transaction* (LOCAL_TRANSACTION) which is a transaction to a resource's local transaction mechanism.

Implicit and Explicit Transactions

Implicit transactions are automatically handled by the Integration Server's transaction manager. When you define an explicit transaction, you define the start-on-completion boundaries of the transaction. As such, implicit and explicit transactions need to be created and managed differently.

The following sections describe implicit and explicit transactions and how to manage them.

Implicit Transactions

With implicit transactions, the Integration Server automatically manages local transactions without requiring you to explicitly do anything. That is, the Integration Server starts and completes an implicit transaction with no additional service calls required by the adapter user.

A transaction context, which the transaction manager uses to define a unit of work, starts when an adapter service is encountered in a flow service execution. The connection required by the adapter service is registered with the newly created context and used by the adapter service. If another adapter service is encountered, the transaction context is searched to see if the connection is already registered. If the connection is already registered, the adapter service uses this connection. If the connection is not registered, a new connection instance is retrieved and registered with the transaction.

Note that if the top level flow invokes another flow, adapter services in the child flow use the same transaction context.

When the top level flow completes, the transaction is completed and is either committed or rolled back, depending on the status (success or failure) of the top level flow.

A single transaction context can contain no more than one LOCAL_TRANSACTION connection. If your flow contains adapter services that use more than one LOCAL_TRANSACTION connection, you must use explicit transactions, which are described in the next section.

For information about designing and using flows, see the *webMethods Developer User's Guide*. For more information about transaction types, see [“Transaction Management of Oracle Applications Adapter Connections” on page 15](#).

Explicit Transactions

You use explicit transactions when you need to explicitly control the transactional units of work. To do this, you use additional services, known as built-in services, in your flow.

A transaction context starts when the `pub.art.transaction.startTransaction()` service is executed. The transaction context is completed when either the `pub.art.transaction.commitTransaction()` or `pub.art.transaction.rollbackTransaction()` service is executed. As with implicit transactions, a single transaction context can contain no more than one LOCAL_TRANSACTION connection.



Note: With explicit transactions, you must be sure to call either a `commitTransaction()` or `rollbackTransaction()` each `startTransaction()`; otherwise you will have dangling transactions which will require you to reboot the Integration Server.

A new explicit transaction context can be started within a transaction context, provided that you ensure that the transactions within the transaction context are completed in the reverse order they were started—that is, the last transaction to start should be the first transaction to complete, and so forth.

For example, consider the following is a valid construct:

```
pub.art.transaction.startTransaction()
    pub.art.transaction.startTransaction()
        pub.art.transaction.startTransaction()
        pub.art.transaction.commitTransaction()
    pub.art.transaction.commitTransaction()
pub.art.transaction.commitTransaction()
```

The following example shows an *invalid* construct:

```
pub.art.transaction.startTransaction()
    pub.art.transaction.startTransaction()
pub.art.transaction.commitTransaction()
    pub.art.transaction.commitTransaction()
```

For information about designing and using flows, see the *webMethods Developer User's Guide*. For more information about transaction types, see [“Transaction Management of Oracle Applications Adapter Connections” on page 15](#).

Changing the Integration Server's Transaction Timeout Interval

The Integration Server's default transaction timeout is no timeout (NO_TIMEOUT). To change the server's transaction timeout interval, use a text editor to modify the `server.cnf` file and add the parameter below. Note that this parameter does not exist by default in the `server.cnf` file; you must add it to the file as described below.

Be sure to shut down the Integration Server before you edit this file. After you make changes, restart the server.

Add the following parameter to the `server.cnf` file:

```
watt.art.tmgr.timeout=TransactionTimeout
```

where *TransactionTimeout* is the number of seconds before transaction timeout.

This transaction timeout parameter does not halt the execution of a flow; it is the maximum number of seconds that a transaction can remain open and still be considered valid. For example, if your current transaction has a timeout value of 60 seconds and your flow takes 120 seconds to complete, the transaction manager will rollback all registered operations regardless of the execution status.

See the *webMethods Integration Server Administrator's Guide* for more information about adding parameters to the `server.cnf` file.

Index

A

- Access Control Lists, assigning 27
- ACL usage to control adapter service access 27
- ACLs, assigning 27
- adapater connections
 - disabling 43
- adapter architecture and components 10
- adapter connection templates 11
- adapter connections
 - enabling 43
- adapter connections. See connections.
- adapter notification templates 12
- adapter notifications. See notifications.
- adapter package
 - loading and unloading 26
 - management guidelines 24
- adapter service templates 12
- adapter services. See services.
- Automatic data validation icon 81
- Automatic polling of adapter metadata option 71, 81

B

- boundary, transaction 15
- built-in services
 - for adapter services 16
 - for connections 16
- business-process integration
 - example of 13

C

- clustering
 - benefits 28
 - cluster store 28
 - connection pooling enabled 31
 - defined 28
 - description of 28
 - failover support 28
 - load balancing 28

- replicating packages in clustered environment 28
- requirements 30
- scalability 28
- configuring
 - connections 35
 - notifications 75
 - services
 - Compound Select SQL 65
 - Insert SQL 64
 - Simple SQL 66
- connections
 - built-in services for 16
 - changing at design time 16
 - configuring 35
 - copying 41
 - deleting 42
 - description of 14
 - editing 40
 - preventing use of 42
 - transaction management overview 15
 - transaction types 15
 - viewing 40
- conventions used in this document 7
- creating
 - connections 40

D

- deleting
 - connections 42
 - notifications 80
 - services 70
 - transaction definitions 54
- dependencies
 - package dependencies of namespace node packages 25
- disabling
 - notifications before deleting 80
 - packages 26
 - polling notification 77

- disabling redirection of administrative services in clustered environment 29
- documentation
 - additional 8
 - conventions used 7
 - feedback 8
- drivers
 - database connection management 14

E

- enabling
 - namespace node packages 26
 - packages 26
 - polling notification 77
- example
 - of business-process integration 13
- explicit transactions 99
- exporting
 - configured adapter notifications 78
 - transaction definitions 48

G

- group access control of adapter services 27
- group access control, assigning 27

I

- implicit transactions 15, 98
- importing
 - transaction definitions 46

L

- loading
 - adapter packages 26
 - namespace node packages 26
 - WmART package 26
- LOCAL_TRANSACTION transaction type
 - defined 15
- logging messages for the adapter 84

M

- Manager (webMethods), using with the adapter 21
- message logging for the adapter 84

N

- namespace node packages
 - enabling and disabling 26
 - package dependencies 25
- namespace node packages loading and unloading 26

O

- open interface tables
 - deleting from transaction definitions 53
- Oracle Applications Adapter
 - description of 10
 - package description 13
 - package name 11
- Oracle Applications transaction definitions
 - creating IS-to-Oracle Applications 50
 - creating Oracle Applications-to-IS 51
 - deleting 54
 - description of 17
 - exporting 48
 - importing 46
 - updating IS-to-Oracle Applications 52
 - updating Oracle Applications-to-IS 53
 - working with SQL trees, Oracle-to-IS transactions 55

P

- package dependencies
 - Adapter Services considerations 27
 - of namespace node packages 25
- packages
 - dependencies, setting 27
 - disabling 26
 - enabling 26
 - loading and unloading 26
 - Oracle Applications Adapter
 - exporting 48
 - replicating in clustered environment 28
- packages, adapter
 - loading and unloading 26
- packages, namespace node
 - enabling and disabling 26
 - loading and unloading 26
 - package dependencies 25

perspectives

Test 22

perspectives, in Developer

Details 22

Edit 22

polling notifications, managing 76

predefined transaction services

description of 20

transaction definitions for 17

preventing use of

connections 42

program code conventions in this document 7

R

Reload values from the adapter icon 71

replicating packages in a clustered environment 28

S

scheduling polling notifications 77

services

built-in services for 16

defined 18

editing 69

Reload values from the adapter icon 71

reloading adapter values 71

templates, define 19

testing

changing Developer to the Test perspective 22

validating data, setting option 81

services, built-in

see built-in services

setAdapterServiceNodeConnection service 16

setPollingNotificationNodeConnection service 16

setting up

connections 35, 40

SQL trees

example 56

SQL SELECT statement syntax 55

stored procedure syntax 56

working with 55

T

testing adapter services

changing Developer to the Test perspective, in Developer 22

testing services 68

transaction boundary 15

transaction definitions

creating IS-to-Oracle Applications 50

creating Oracle Applications-to-IS 51

deleting 54

description of 17

exporting 48

for predefined transaction services 17

importing 46

updating IS-to-Oracle Applications 52

updating Oracle Applications-to-IS 53

working with SQL trees, Oracle-to-IS transactions 55

transaction services, predefined

description of 20

Oracle Applications versions supported 20

transaction definitions for 17

transaction support

explicit transactions 99

implicit transactions 98

local transactions 98

transaction types

LOCAL_TRANSACTION 15

troubleshooting information 8

typographical conventions in this document 7

U

unloading

adapter packages 26

namespace node packages 26

WmART package 26

updating

transaction definitions (IS-to-Oracle Applications) 52

transaction definitions (Oracle Applications-to-IS) 53

V

Validate service/notification values icon 81

viewing

 notifications 79

viewing services 69

W

webMethods Manager, using with the adapter 21

WmART package

 and adapter messages and exceptions 85

 built-in services 16

 dependencies 25

 description of 11

 load and reload order 26

 load and unload order 24

 loading and unloading 26

 transaction management and built-in services 98

 unload order 26