webMethods

webMethods JDBC Adapter

User's Guide

VERSION 6.0.3

webMethods, Inc.
South Tower
3877 Fairfax Ridge Road
Fairfax, VA 22030
USA
703.460.2500
http://www.webmethods.com

Document ID: ADAPTER-JDBC-UG-603-20060704

Contents

About This Guide

This guide describes how to configure and use the webMethods JDBC Adapter Version 6.0.3. It contains information for administrators and application developers who want to exchange data with relational databases using a JDBC driver.

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

- JDBC drivers
- Database concepts and basic SQL
- 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.
Italic	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 *folder.subfolder:service*.
`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:

- 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. Be sure to check for any updates or additional information about the JDBC Adapter.

webMethods

CHAPTER 1

Overview of the Adapter

About the JDBC Adapter

The webMethods JDBC Adapter is an add-on to the webMethods Integration Server that enables you to exchange data with relational databases through the use of a JDBC driver. The adapter provides seamless and real-time communication with the database without requiring changes to your existing application infrastructure.

Using the JDBC Adapter, webMethods Integration Server clients can create and run services that execute transactions to retrieve data from, and insert and update data in, relational databases.

For example, you can use the JDBC Adapter to add a customer to an Oracle database based on data from another system connected to the Integration Server. Or you can use the JDBC Adapter to poll a Microsoft SQL Server database for customers that have been added to the database, and to send that data to the Integration Server to be inserted into another resource.

The JDBC Adapter supports the following databases:

- DB2 for AS/400
- DB2 for OS/390
- DB2 Universal Database (UDB)
- IBM Informix
- Microsoft SQL Server
- Oracle
- Sybase
- Teradata

For a list of the database versions, JDBC drivers, and platforms that the JDBC Adapter supports, see the *webMethods JDBC Adapter Installation Guide*.

For a list of known driver limitations, see "Database Driver Known Limitations" on page 187.

In addition to the JDBC Adapter, webMethods also provides the WmDB package that you can use to connect to databases. However, the JDBC Adapter provides more functionality, improved performance, and supports adapter notifications. To help you determine whether you should use the WmDB package or the JDBC Adapter, see Appendix D, "Determining Whether to Use the WmDB Package or the JDBC Adapter". For more information about the WmDB package, see the WmDB user's guide.

Architectural Overview

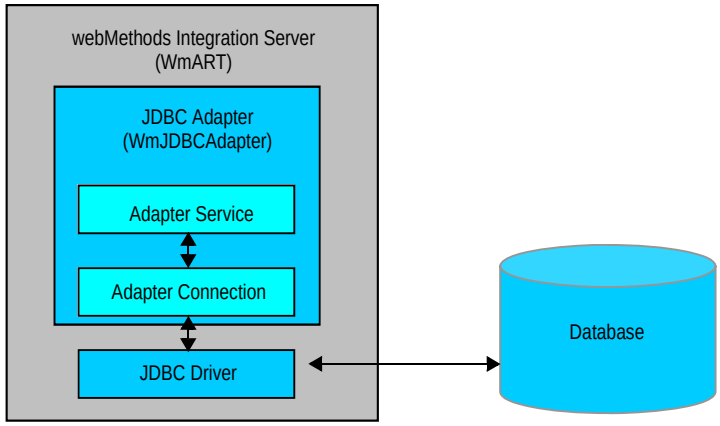
The JDBC Adapter provides a set of user interfaces, services, and templates that enable you to create integrations with databases using a JDBC driver. The adapter is provided as a single package that must be installed on the webMethods Integration Server. For detailed installation instructions and software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

Because the JDBC Adapter uses a JDBC driver to perform operations on databases, the adapter requires a supported JDBC driver to be installed and loaded in the packages directory of the Integration Server. See "JDBC Adapter Package Management" on page 46 for more details.

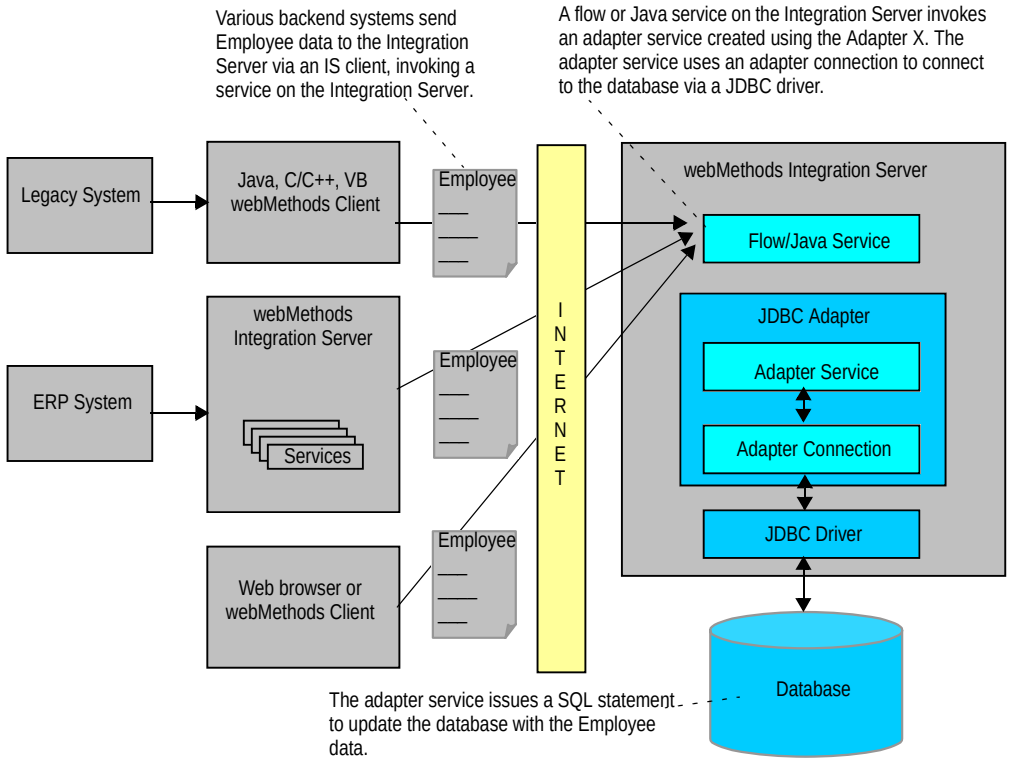
The JDBC Adapter enables you to configure the following components:

- Adapter connections: Enable the Integration Server to connect to database systems at run time. You must configure an adapter connection before you can configure adapter services or adapter notifications. See "Adapter Connections" on page 13 for a detailed description of adapter connections.
- Adapter services: Enable the Integration Server to initiate and perform database operations on a database. For example, an adapter service could enable a trading partner to query your inventory database to determine whether a particular item is currently in stock. You configure adapter services using adapter services templates, which are provided with the JDBC Adapter. See "Adapter Services" on page 17 for a detailed description of adapter services.
- Adapter notifications: Monitor a database and notify the Integration Server when an action (not initiated by the Integration Server) has occurred on a particular database table. For example, an adapter notification could notify the Integration Server when an update operation was performed on a particular database table. See "Adapter Notifications" on page 22 for a detailed description of adapter notifications.

The following diagram shows at a high level how an adapter service uses an adapter connection and a JDBC driver to connect to and perform an operation on a database.



The next diagram shows a business integration where an adapter service is used to update a database with employee data. The employee data could be provided by several different types of external Integration Server (IS) clients.



The architecture for integrations using adapter notifications is similar to the architecture for integrations using adapter services shown above, but it varies according to the type of notification. The primary difference between these types of integrations is that notifications are initiated by events that occur on the database, not by actions that occur on the Integration Server.

With adapter notifications, you can capture event data from the database and use it to initiate another action within the Integration Server. For example, you could create an adapter notification to monitor an employee table within a database and whenever an employee is added to the table, you could post that employee data to a webMethods Broker. webMethods Broker clients could then subscribe to that notification's publishable document.

See "Adapter Notifications" starting on page 22 for more information about the architecture for the different types of adapter notifications.

Package Management

The JDBC Adapter is provided as a package called WmJDBCAdapter that you manage like any package on the Integration Server.

There are several considerations regarding how you set up and effectively manage your packages on the Integration Server:

- You must create user-defined packages for your connections, adapter services, and notifications. See "JDBC Adapter Package Management" on page 46 for details.
- You should understand how package dependencies work so you make the best decisions regarding how you manage your adapter services and notifications. See "Package Dependency Requirements and Guidelines" on page 47 for details.
- You control which development groups have access to which adapter services and notifications. See "Group Access Control" on page 49 for details.
- You should understand how clustering, an advanced feature of the webMethods Integration Server, works to effectively manage your adapter services. See "JDBC Adapter in a Clustered Environment" on page 49 for details.

Adapter Connections

The JDBC Adapter connects to a database through a JDBC driver at run time. You create one or more connections at design time to use in integrations. The number of connections you create, and the types of those connections, depend on the types of databases you are connecting to and your integration needs. For example, if you are connecting to an Oracle database and a DB2 Server database, you will need to create connections that are unique to those two databases. Additionally, if you have multiple installations of the same kinds of databases, you access each using different connections. For example, if you have a data

warehouse system and an ERP system that use your Oracle database, you create a connection for each system.

JDBC Adapter connections contain parameters that the Integration Server uses to manage connections to the database so that they can be used by the JDBC Adapter to provide services. You configure connections using the webMethods Integration Server Administrator tool. You must have webMethods administrator privileges to access the JDBC Adapter's administrative screens.

For instructions for configuring, viewing, editing, enabling, and disabling JDBC Adapter connections, see "JDBC Adapter Connections" on page 65. For information about setting user privileges, see the *webMethods Integration Server Administrator's Guide*.

For a list of tasks that you must do before you can create your connections, see "Before Configuring or Managing Adapter Connections" on page 66.

Using JDBC Drivers to Connect to Databases

JDBC Adapter connections access databases using either the driver's DataSource or XADataSource objects provided by your JDBC driver. For more information about DataSource and XADataSource objects, see the documentation provided with your JDBC driver.

The JDBC driver must be installed on the Integration Server machine and loaded when the server starts. See "Installing the JDBC Driver on the Integration Server" on page 67 for instructions.

Note: If you plan to use CLOBs or BLOBs that exceed 4 KB, be sure to use an Oracle OCI driver or Oracle 10g Thin driver.

For more information about the transaction types supported in the JDBC Adapter, see "Transaction Management of JDBC Adapter Connections" on page 15.

For a complete list of the JDBC drivers that the adapter supports, see the *webMethods JDBC Adapter Installation Guide*.

For a list of known driver limitations, see "Database Driver Known Limitations" on page 187.

Transaction Management of JDBC Adapter Connections

JDBC Adapter connections support the following transaction types:

Transaction Type	Description
NO_TRANSACTION	The connection provides no transaction control over the operations being performed. That is, the connection automatically commits (Auto Commit) all operations.
LOCAL_TRANSACTION	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.
XA_TRANSACTION	This transaction type allows the connection to support two-phase transactions executed across multiple databases. In one transaction boundary, all of the operations on multiple connections 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. Note: All of the connections involved in a two-phase transaction must support the XA_TRANSACTION transaction type.

Note: Insert Notifications, Update Notifications, Delete Notifications and Ordered Notifications support LOCAL_TRANSACTION mode only.

When you define 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 transaction manager.

You can also explicitly manage transactions using built-in services. See Appendix B, "Built-In Transaction Management Services", 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.

Runtime 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 or later, 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, "JDBC Adapter Connections", on page 65.

Built-In Services For Connections

Beginning with version 6.1, the Integration Server 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 or 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 19.

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

Adapter Services

To use the JDBC Adapter, you create adapter services. Adapter services allow you to connect to the adapter's resource and initiate an operation on the resource from the Integration Server.

You call adapter services from flow or Java services to interact with database tables. The adapter services perform database operations by calling JDBC APIs. The Integration Server then uses adapter connections that you defined earlier to execute the adapter services. See "Adapter Service Transaction Processing" on page 21 for details.

Adapter services are based on templates provided with the JDBC Adapter. Each template represents a specific technique for doing work on a resource, such as using the SelectSQL template to retrieve specified information from a database.

An adapter service template contains all the code necessary for interacting with the resource but without the data specifications. You provide these specifications when you create a new adapter service.

Creating a new service from an adapter service template is straightforward. Using the webMethods Developer, you assign the service a default adapter connection.

After you select the connection for the adapter service, you select the adapter service template and supply the data specifications using the webMethods Developer. Some familiarity with using the webMethods Developer is required. See the *webMethods Developer User's Guide* for more information.

The JDBC Adapter provides the following adapter service templates:

Adapter Service Type	Adapter Service Template	Description
Select SQL	SelectSQL	Retrieves specified information from a database table. See "Configuring SelectSQL Services" on page 89 for instructions.
Insert SQL	InsertSQL	Inserts new information into a database table. See "Configuring InsertSQL Services" on page 92 for instructions.
Update SQL	UpdateSQL	Updates existing information in a database table and includes a mapping for an output field that stores the number of affected rows. See "Configuring UpdateSQL Services" on page 95 for instructions.

Adapter Service Type	Adapter Service Template	Description
Batch Insert SQL	BatchInsertSQL	Inserts new information into a database table. Use this service when you will be inserting a large volume of data into a single table. The data source for the service can be from a flat file or from other services that generate an Integration Server document list. See "Configuring BatchInsertSQL Services" on page 99 for instructions.
Batch Update SQL	BatchUpdateSQL	Updates information in a database table when dealing with a large volume of data. Use this service when you will be updating a large volume of data in a single table. The data source for the service can be from a flat file or from other services that generate an Integration Server document list. See "Configuring BatchUpdateSQL Services" on page 102 for instructions.
Delete SQL	DeleteSQL	Deletes rows from a table and includes a mapping for an output field that stores the number of affected rows. See "Configuring DeleteSQL Services" on page 106 for instructions.
Custom SQL	CustomSQL	Defines and executes custom SQL to perform database operations. You can execute almost any SQL statement required by integrations, such as data management statements. See "Configuring CustomSQL Services" on page 109 for instructions.
Dynamic SQL	DynamicSQL	Defines and executes a SQL statement, part of which you set at run time through the input field. See "Configuring DynamicSQL Services" on page 111 for instructions.
Stored Procedure	StoredProcedure	Calls a stored procedure to perform database operations. See "Configuring StoredProcedure Services" on page 115 for instructions.

Using Adapter Services

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

	For this task...	Use these tools...
1	Create an adapter connection. See "JDBC Adapter Connections" on page 65 for details.	Integration Server Administrator
2	Select the appropriate adapter service template and configure the adapter service. Depending on the type of adapter service, you specify the: ■ Adapter connection ■ Database table ■ SQL expression used to modify or select data, or a stored procedure to execute against the database (for StoredProcedure Adapter Services) ■ Input field(s) and type(s) as needed ■ Output field(s) and type(s) as needed See "Adapter Services" on page 87 for more information about configuring adapter services.	Developer
3	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.	Developer
4	Manage the adapter service. See "Package Management" on page 45, "Adapter Services" on page 87, and "Logging and Exception Handling" on page 157 for details.	Developer and Integration Server Administrator

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

When using versions of the Integration Server earlier than 6.1 you may *not* assign a different connection to an adapter service or notification after you configure the adapter service. (You may, however, change the parameters of the connection.) If you want to use a different connection, you must create 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 database servers, you must configure multiple, separate adapter services, one for each connection you use.

Beginning with version 6.1, the Integration Server solves this limitation by providing 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 this function, you can change the specific connection associated with an adapter service at design time so that you do not need to create and maintain multiple adapter services.

Note: This built-in service can be run at design time only; do not use it 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.

Changing the Connection Associated with an Adapter Service at Runtime

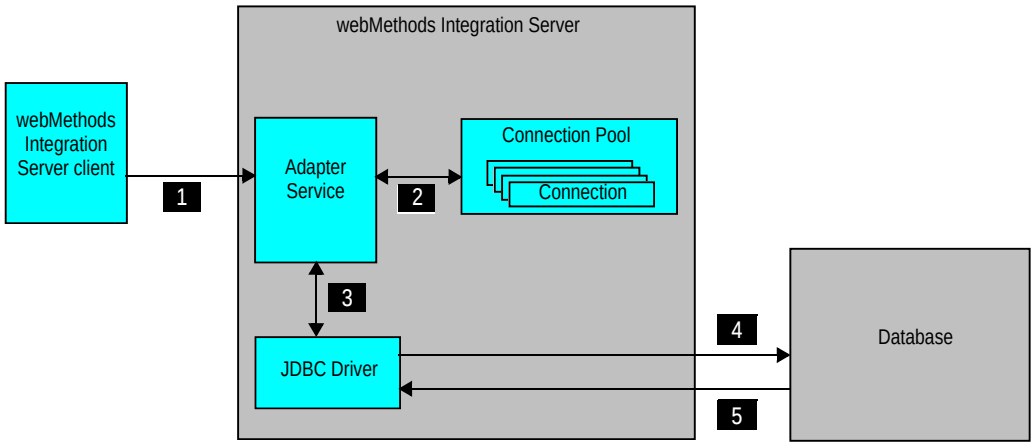
Beginning with version 6.5, the Integration Server enables you to dynamically select the connection a service uses to interact with the adapter's resource. This feature enables one service to interact with multiple, similar backend resources.

For example, a service can be defined to use a default connection that interacts with your company's production database. However, at runtime you can override the default connection and instead use another connection to interact with the company's test database.

For more information about overriding a service's default connection at runtime, see "Dynamically Changing a Service's Connection at Runtime" on page 81.

Adapter Service Transaction Processing

The following diagram illustrates how the JDBC Adapter processes adapter services at run time.



Step	Description
1	An Integration Server client, typically using a flow or Java service, invokes a JDBC Adapter service on the webMethods Integration Server to perform an operation on a database. You configured the adapter service earlier using the webMethods Developer.
2	The adapter service gets a connection from the service's connection pool. Adapter connections contain connection information for the database, including JDBC driver parameters.
3	The adapter service uses the JDBC driver to connect to the database. You created and enabled the adapter connection earlier using the Integration Server Administrator.

Step	Description
4	The adapter service performs a SQL operation against the database. ■ For SelectSQL, InsertSQL, UpdateSQL, DeleteSQL, CustomSQL, and DynamicSQL services, the adapter service executes a SQL statement against the database. ■ For BatchInsertSQL and BatchUpdateSQL services, the adapter service executes batch SQL statements against the database. The adapter service will continue to loop through the document list that is used as input, set the fields to the parameters of the SQL statement and then add that command set to the batch. Upon completion, the adapter sends the entire batch to the database resource for execution. ■ For StoredProcedure services, the adapter service executes a stored procedure against the database.
5	Depending on the adapter service type, such as a SelectSQL service, the adapter service may return data to the Integration Server. ■ If the operation is successful, the service returns the output from the service's database operation, if applicable. With BatchInsertSQL and BatchUpdateSQL services, if all commands are successfully executed, the adapter commits all commands in the batch and returns a list of String values. These values will vary by driver; refer to your driver documentation for details. ■ If the operation is unsuccessful, the service returns an error such as an AdapterException. If the database throws an exception while performing the adapter service's operation, the adapter passes the exception to the Integration Server logs. For more information about how the adapter handles exceptions, see "JDBC Adapter Exception Handling" on page 159.

Adapter Notifications

An adapter notification monitors a specified database table for changes, such as an insert, update, or delete operation, so that the appropriate Java or flow services can make use of the data, such as sending an invoice or publishing it to the Integration Server.

JDBC Adapter notifications are polling-based. That is, the Integration Server will invoke the notification periodically based on the polling interval that you specify when you schedule the notification as described in "Managing Polling Notifications" on page 149.

Adapter notifications vary somewhat in how they work, depending on the type of the adapter notification. Be sure to review "Notification Types" beginning on page 24 to understand how their operations differ.

Adapter Notification Templates

The JDBC Adapter provides the following adapter notification templates:

Notification Type	Notification Template	Description
Insert Notification	InsertNotification	Publishes notification of insert operations on a database table. See "Configuring InsertNotifications" on page 125 for instructions.
Update Notification	UpdateNotification	Publishes notification of update operations on a database table. See "Configuring UpdateNotifications" on page 129 for instructions.
Delete Notification	DeleteNotification	Publishes notification of delete operations on a database table. See "Configuring DeleteNotifications" on page 134 for instructions.
Basic Notification	BasicNotification	Polls a database table for data using a SQL Select operation. See "Configuring BasicNotifications" on page 138 for instructions.
Stored Procedure Notification	StoredProcedure Notification	Publishes notification data by calling a stored procedure inside of a database. See "Configuring StoredProcedureNotifications" on page 142 for instructions.
Ordered Notification	OrderedNotification	Publishes notification data for multiple insert, update, or delete operations on multiple tables for a given database. See "Configuring OrderedNotifications" on page 145 for instructions.

Exactly Once Notification Feature

Most adapter notifications, such as Insert Notifications and Update Notifications, can use the Exactly Once notification feature. This feature ensures that notification data will not be duplicated even if a failure occurs during processing. This is achieved by assigning unique IDs for each publishable document. After a processing failure, the Integration Server checks for duplicate records in storage and ignores any duplicate IDs.

Note: Stored Procedure Notifications do not support the Exactly Once notification feature because they do not use publishable document unique IDs.

See "Using the Exactly Once Notification Feature" on page 152 for more details.

Notification Types

There are six types of notifications: Insert, Update, Delete, Basic, Stored Procedure, and Ordered Notifications. They vary in how they are structured and operate, as described in the following sections.

Insert Notifications, Update Notifications, and Delete Notifications

Insert Notifications, Update Notifications, and Delete Notifications use a combination of triggers and buffer tables to capture events that happen on specific tables in a database. You configure the triggers and buffer tables when you configure the notifications.

These types of notifications operate similarly, with the exception of the type of SQL operation (insert, update, or delete) that they monitor. The adapter creates the trigger and buffer table when you enable a notification. The buffer table, which you specified when you configured the notification, holds the data selected by the trigger. There are no special size constraints for the buffer tables. The trigger monitors the database table you specified when you configured the notification and inserts data into the buffer table. When you disable a notification, the adapter drops the trigger and buffer table.

When you enable a notification, the database trigger monitors the table and inserts the data into the buffer table. When the Integration Server invokes the notification, it retrieves the rows of data from the buffer table, publishes each row in the notification's publishable document, and then removes this row from the buffer table.

After you enable these types of notifications, the trigger and buffer table remain in the database table when you:

- Shut down the Integration Server
- Disable the package containing the enabled notification
- Reload the package containing the enabled notification
- Suspend the notification (Integration Server 6.5)

In the meantime, the trigger continues to monitor the table and to insert data into the buffer table. The Integration Server invokes the enabled notification when it restarts, or when it enables or reloads the package that contains this notification. See "Insert, Update, and Delete Notifications Transaction Processing" on page 27 for more information about how these types of notifications work.

See "Configuring InsertNotifications" on page 125, "Configuring UpdateNotifications" on page 129, or "Configuring DeleteNotifications" on page 134 for instructions for configuring this type of adapter notification.

For more details about the 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 or later, see the *Publish-Subscribe Developer's Guide*.

Using Insert, Update, and Delete Notifications

The following table lists the tasks required to use these types of notification:

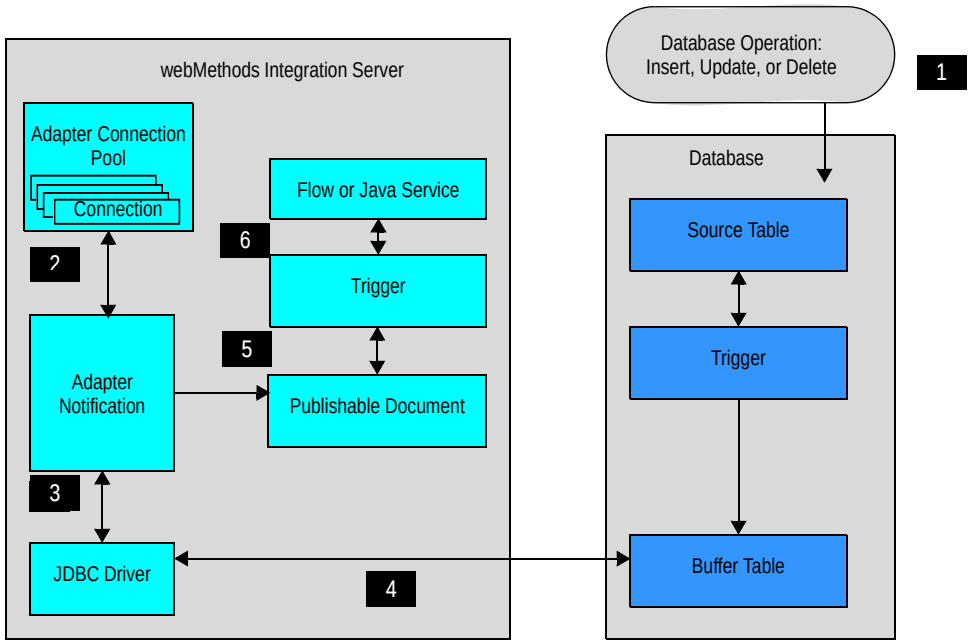
	For this task...	Use these tools...
1	Create an adapter connection. See "Configuring JDBC Adapter Connections" on page 68 for details.	Integration Server Administrator
2	Configure the notification and specify the: ■ Adapter connection ■ Source table ■ Publishable document to contain the data from the buffer table. There is a single publishable document used for all events associated with the notification. For more details about the 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 or later, see the *Publish-Subscribe Developer's Guide*. ■ Output data fields contained in the publishable document ■ Database trigger and buffer table See "Adapter Notifications" on page 123 for instructions for configuring notifications.	Developer

For this task...		Use these tools...
3	If you plan to use an Integration Server flow or Java service, design it to react to the data changes contained in the notification's publishable document. Create the Integration Server trigger to use the notification's publishable document. See the *webMethods Developer User's Guide* for details.	Developer
4	Schedule and enable the adapter notification. When you enable the notification: ■ It automatically creates the database trigger and buffer table you configured when you created the notification. ■ The Integration Server Scheduler invokes the notification and continues to do so periodically, based on the polling schedule parameters you created earlier. See "Managing Polling Notifications" on page 149 for instructions for scheduling and enabling notifications.	Integration Server Administrator
5	Manage the notification. See "Package Management" on page 45, "Adapter Notifications" on page 123, and "Logging and Exception Handling" on page 157 for details.	Developer and Integration Server Administrator

Insert, Update, and Delete Notifications Transaction Processing

The following diagram and steps illustrate what happens when these types of notifications are invoked. The Integration Server continues to invoke the notification periodically, as defined when you configured the schedule parameters for polling the notification.

Insert, Update, and Delete Notifications



Step	Description
1	Insert Notifications, Update Notifications, and Delete Notifications monitor an operation that happens to a database table, such as an insert, update, or delete operation. You specified the source table to monitor at the time you configured the adapter.
2	The notification gets a connection from the service's connection pool. Adapter connections contain connection information for the database, including JDBC driver parameters.
3	The notification uses the JDBC driver to connect to the database. You created and enabled the adapter connection earlier using the Integration Server Administrator.

Step	Description
4	The notification retrieves the rows of data from the buffer table. The buffer table holds the data selected by the trigger. While the adapter remains enabled, the trigger continues to monitor the database table and insert data into the buffer table.
5	The notification creates the publishable document, which contains a row of data from the buffer table. The notification publishes the publishable document. For more details about the 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 or later, see the *Publish-Subscribe Developer's Guide*.
6	Using an Integration Server trigger you configured to use the notification's publishable document, a flow or Java service on the Integration Server is invoked to react to the data changes contained in the publishable document. After the data is published, the data is removed from the buffer table.

Basic Notifications

In contrast with Insert Notifications, Update Notifications, and Delete Notifications, Basic Notifications require you to define a buffer table, and a database trigger or other means of monitoring database changes so that changes are written into the buffer table.

To monitor database changes, a Basic Notification queries the buffer table. Basic Notifications provide you with the flexibility to manage buffer tables, such as a table with user privileges, and to tailor your own database monitoring methods for producing notification data. By default, after the data is retrieved and processed, it is deleted from the buffer table to ensure that the data is not processed multiple times. If you do not choose the default, that is, if you do no not allow the adapter to delete data from the buffer table, be sure that you use some other means to purge the data so that data does not get published more than once.

See "Basic Notifications Transaction Processing" on page 30 for more information about how Basic Notifications work.

See "Configuring BasicNotifications" on page 138 for instructions for configuring this type of adapter notification.

Using Basic Notifications

The following table lists the tasks required to use this notification:

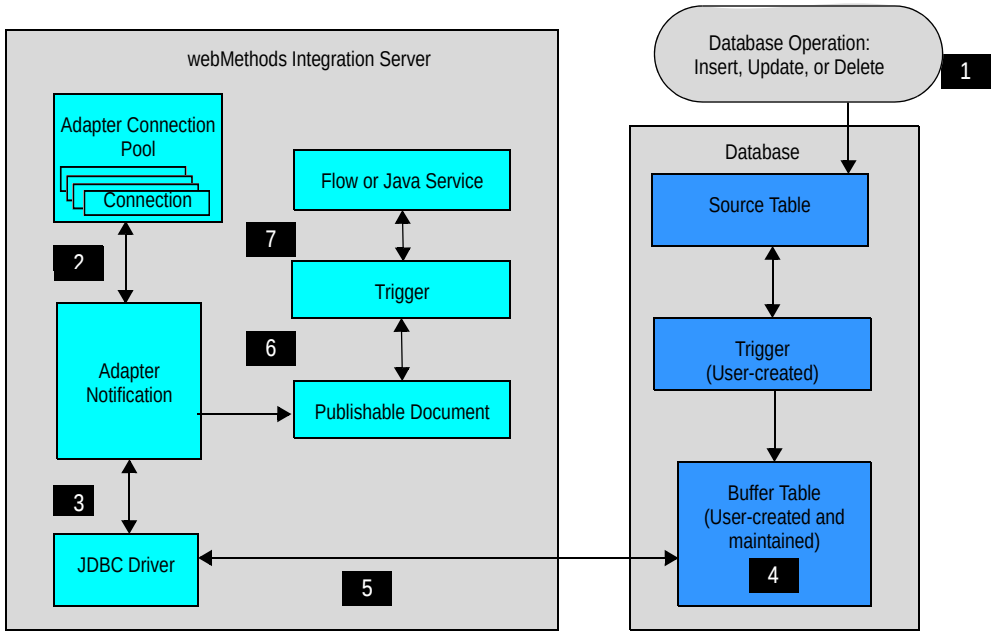
	For this task...	Use these tools...
1	If needed, create your own buffer table and database trigger (or other means) to monitor for database changes.	User-defined
2	Create an adapter connection. See "Configuring JDBC Adapter Connections" on page 68 for details.	Integration Server Administrator
3	Configure the notification and specify the: ■ Adapter connection ■ Buffer tables that you created independently ■ Publishable document to contain the data from the buffer table. There is a single publishable document used for all events associated with the notification. For more details about the 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 or later, see the *Publish-Subscribe Developer's Guide*. ■ Output data fields contained in the publishable document See "Configuring BasicNotifications" on page 138 for instructions for configuring this type of notification.	Developer
4	If you plan to use an Integration Server flow or Java service, design it to react to the data changes contained in the notification's publishable document. Create the Integration Server trigger to use the notification's publishable document. See the *webMethods Developer User's Guide* for details.	Developer
5	Schedule and enable the adapter notification. When you enable the notification, the Integration Server Scheduler invokes the notification periodically and continues to do so, based on the polling schedule parameters you created earlier. See "Managing Polling Notifications" on page 149 for instructions for scheduling and enabling notifications.	Integration Server Administrator

For this task...		Use these tools...
6	Manage the notification. See "Package Management" on page 45, "Adapter Notifications" on page 123, and "Logging and Exception Handling" on page 157 for details.	Developer and Integration Server Administrator

Basic Notifications Transaction Processing

The following diagram and steps illustrate what happens when a Basic Notification is invoked. The Integration Server continues to invoke the notification periodically, as defined when you configured the polling schedule parameters for the notification.

Basic Notifications



Step	Description
1	Basic Notifications monitor an operation that happens to a database table, such as an insert, update, or delete operation. You specified the buffer table to monitor at the time you configured the adapter.
2	The notification gets a connection from the service's connection pool. Adapter connections contain connection information for the database, including JDBC driver parameters.

Step	Description
3	The notification uses the JDBC driver to connect to the database. You created and enabled the adapter connection earlier using the Integration Server Administrator.
4	Unlike Insert Notifications, Update Notifications, and Delete Notifications, you create your own buffer table and trigger, or other means of monitoring database changes. The diagram and steps listed here assume you are creating your own buffer table and trigger to monitor for changes. The buffer table you define will hold the data selected by any trigger you create. The trigger will monitor the database table and insert data into the buffer table.
5	The notification retrieves the rows of data from the buffer table.
6	The notification creates the publishable document, which contains a row of data from the buffer table. The notification publishes the publishable document. For more details about the 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 or later, see the *Publish-Subscribe Developer's Guide*.
7	Using an Integration Server trigger you configured to use the notification's publishable document, a flow or Java service on the Integration Server is invoked to react to the data changes contained in the publishable document. After the data is published, the data in the buffer table will be retained or removed, depending on how you configured your buffer table and trigger.

Stored Procedure Notifications

A Stored Procedure Notification calls a stored procedure you created earlier to publish notification data in the notification's publishable document(s). See "Stored Procedure Notifications Transaction Processing" on page 33 for more information about how Stored Procedure Notifications work.

See "Configuring StoredProcedureNotifications" on page 142 for information about configuring this type of adapter notification.

Note: Stored Procedure Notifications do not support the Exactly Once notification feature because they do not use publishable document unique IDs. (See "Using the Exactly Once Notification Feature" on page 152 for details about this feature.)

Using Stored Procedure Notifications

The following table lists the tasks required to use this notification:

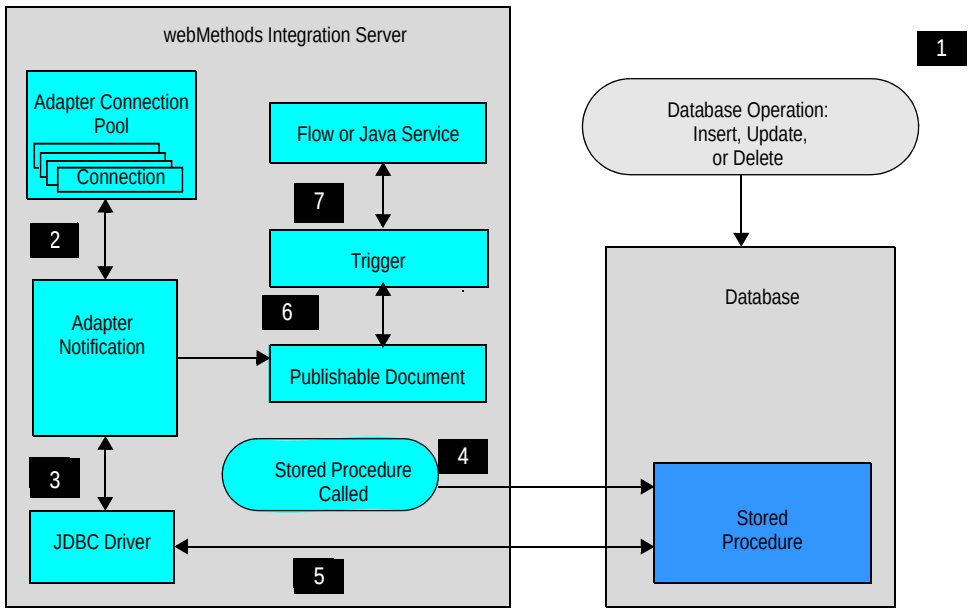
	For this task...	Use these tools...
1	To ensure that the same data is not published multiple times, design and test your stored procedure so that whenever the stored procedure is invoked, you are assured that it provides the correct data.	User-dependent
2	Create an adapter connection. See "Configuring JDBC Adapter Connections" on page 68 for details.	Integration Server Administrator
3	Configure the notification and specify the: ■ Adapter connection ■ Stored procedure ■ Publishable document to contain the data. There is a single publishable document used for all events associated with the notification. For more details about the 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 or later, see the *Publish-Subscribe Developer's Guide*. ■ Any output data fields to be contained in the publishable document See "Configuring StoredProcedureNotifications" on page 142 for instructions for configuring this type of notification.	Developer
4	If you plan to use an Integration Server flow or Java service, design it to react to the data changes contained in the notification's publishable document. Create the Integration Server trigger to use the notification's publishable document. See the *webMethods Developer User's Guide* for details.	Developer

For this task...		Use these tools...
5	Schedule and enable the adapter notification. When you enable the notification, the Integration Server Scheduler invokes the notification and continues to do so periodically, based on the polling schedule parameters you created earlier. See "Managing Polling Notifications" on page 149 for instructions for scheduling and enabling notifications.	Integration Server Administrator
6	Manage the notification. See "Package Management" on page 45, "Adapter Notifications" on page 123, and "Logging and Exception Handling" on page 157 for details.	Developer and Integration Server Administrator

Stored Procedure Notifications Transaction Processing

The following diagram and steps illustrate what happens when a Stored Procedure Notification is invoked.

Stored Procedure Notifications



Step	Description
1	A Stored Procedure Notification uses a stored procedure you created in the database to monitor an operation that happens to a database table, such as an insert, update, or delete operation. When the Stored Procedure Notification calls the stored procedure, it stores any output in the notification's publishable document(s).
2	The notification gets a connection from the service's connection pool. Adapter connections contain connection information for the database, including JDBC driver parameters.
3	The notification uses the JDBC driver to connect to the database. You created and enabled the adapter connection earlier using the Integration Server Administrator.
4	The Integration Server calls the stored procedure.
5	The notification retrieves each row of data from the stored procedure.
6	Each row of data is published using the notification's publishable document. Depending on the stored procedure, the Stored Procedure Notification's publishable document(s) can contain any of the following: ■ Output parameter(s): if the called stored procedure has any output parameters, they are contained in any publishable documents for the Stored Procedure Notification. ■ Return value(s): if the called stored procedure returns any values, then a return value is contained in a publishable document for the Stored Procedure Notification. ■ Single result set (or Oracle REF CURSOR): Stored Procedure Notifications can support one result set. If a call to the stored procedure produces a result set, then the single result set is contained in one or more publishable documents for the Stored Procedure Notification. In some cases, a call to a Stored Procedure Notification can produce a single result set that contains multiple records. In this case, each record will have a separate publishable document, containing one row and one or more columns, that is returned to the adapter. For more details about the 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 or later, see the *Publish-Subscribe Developer's Guide*.

Step	Description
7	Using an Integration Server trigger you configured to use the notification's publishable document, a flow or Java service on the Integration Server is invoked to react to the data changes contained in the publishable document.

Ordered Notifications

You use Ordered Notifications to monitor multiple insert, update, or delete operations on one or more tables for a given database by creating a single notification using the same publishable document. Similar to Insert Notifications, Update Notifications, and Delete Notifications, Ordered Notifications use triggers and buffer tables to capture events that happen on specific tables in a database.

After you enable the Ordered Notification, the trigger, buffer table, and sequence remain in the database table when you:

- Shut down the Integration Server.
- Disable the package containing the enabled Ordered Notification.
- Reload the package containing the enabled Ordered Notification.

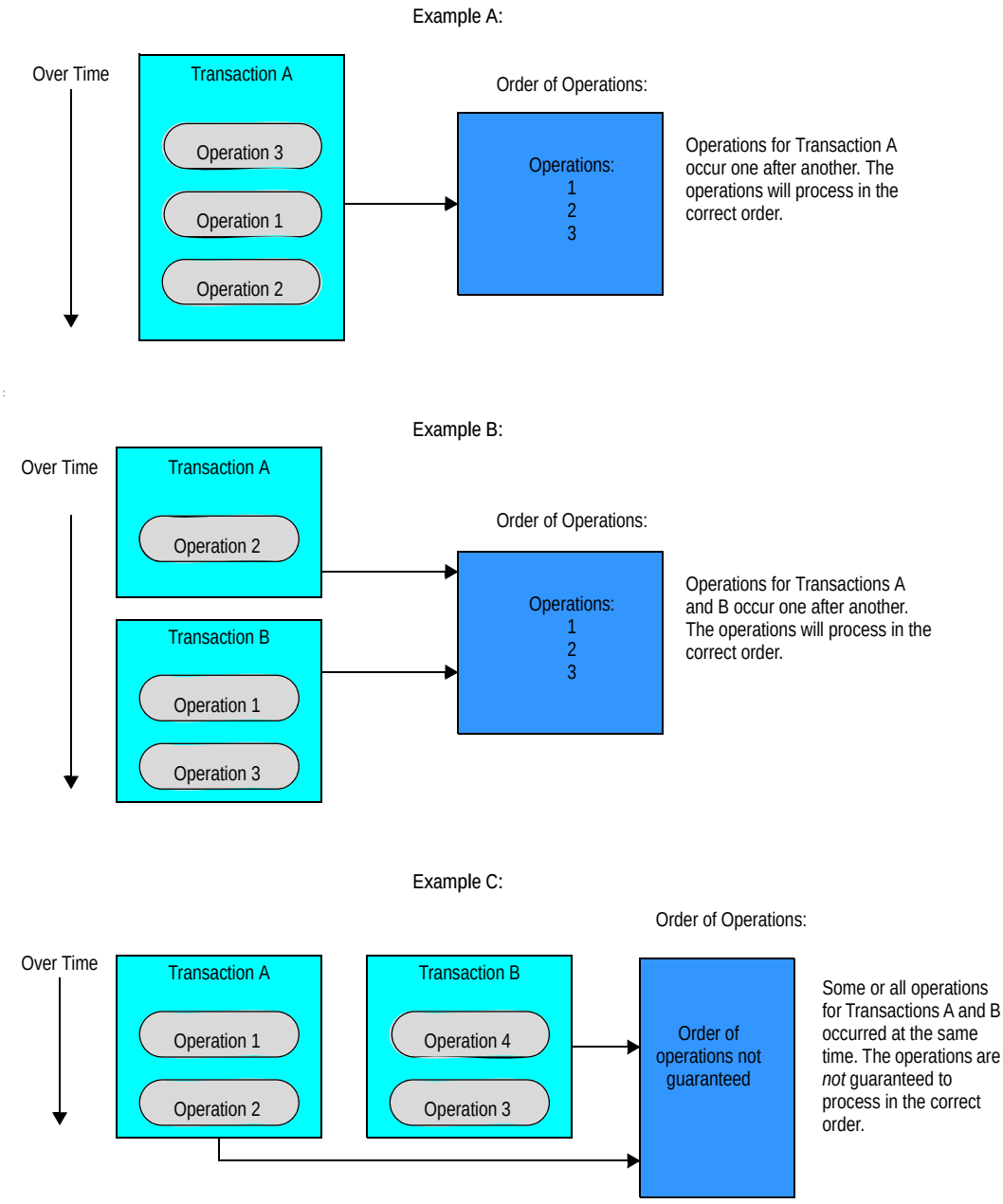
In the meantime, the trigger continues to monitor the table and to insert data into the buffer table. The Integration Server invokes the enabled Ordered Notification when it restarts, or when it enables or reloads the package that contains this notification.

When you disable a notification, the adapter drops the trigger and buffer table.

See "Ordered Notifications Transaction Processing" on page 38 for more information about how Ordered Notifications work.

Considerations when Using Ordered Notifications

- Use the Ordered Notification only if you need to preserve the order in which the operations occur; otherwise, use Insert Notifications, Update Notifications, and Delete Notifications because they have better performance.
- Ordered Notifications ensure that the operations process in the correct order when they occur sequentially in one transaction; however, order preservation is not guaranteed if the operations occur in concurrent transactions. For example, see the following diagrams. Examples A and B will process operations in the correct order. Example C is not guaranteed to process operations in the correct order.



Configuring an Integration Server Trigger and Flow Service

With Ordered Notifications, you typically configure an Integration Server trigger to subscribe to the notification's publishable document and a flow service that the trigger invokes. Because the primary reason to use Ordered Notifications is to preserve the order

in which the operations occur, be sure to use the Process Document Serially option on the Settings tab in the Developer when you create the trigger and flow service.

For more information about using configuring Integration Server triggers and flow services, see the *webMethods Developer User's Guide*.

Using Ordered Notifications

Note: You can create only one trigger for each operation on a table. For each notification, you can configure only one trigger for each table.

The following table lists the tasks required to use this notification:

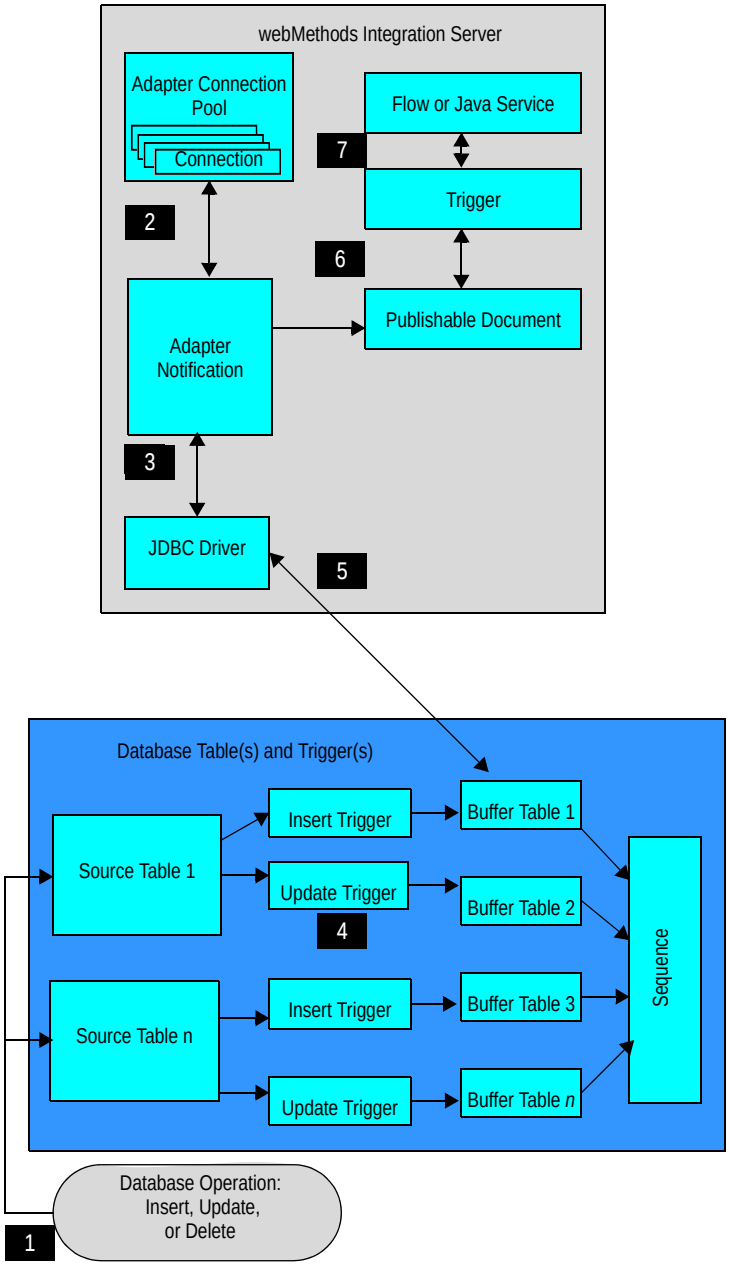
	For this task...	Use these tools...
1	Create an adapter connection. See "Configuring JDBC Adapter Connections" on page 68 for details.	Integration Server Administrator
2	Configure the notification and specify the: ■ Adapter connection ■ Source tables ■ Type of operation associated with the Ordered Notification; that is, an insert, update, or delete operation ■ Operation ID you create for each operation ■ Output data fields to be published for each operation ■ Database trigger and buffer table The buffer table will hold the data selected by the trigger. The trigger will monitor the database table and insert data into the buffer table. See "Configuring an Integration Server Trigger and Flow Service" for more details.	Developer

For this task...		Use these tools...
	■ Publishable document to contain the data from the buffer table. There is a single publishable document used for all events associated with the notification. For more details about the 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 or later, see the *Publish-Subscribe Developer's Guide*. See "Configuring OrderedNotifications" on page 145 for instructions for configuring this type of notification.	
3	If you plan to use an Integration Server flow or Java service, design it to react to the data changes contained in the notification's publishable document. Create the Integration Server trigger to use the notification's publishable document. See the *webMethods Developer User's Guide* for details. If you use a trigger, be sure to enable the Process Document Serially option. See "Configuring an Integration Server Trigger and Flow Service" on page 36 for details.	Developer
4	Schedule and enable the adapter notification. When you enable the notification, it automatically creates the database trigger, sequence, and buffer table you configured when you created the notification. The Integration Server Scheduler invokes the notification and continues to do so. See "Managing Polling Notifications" on page 149 for instruction for scheduling and enabling notifications.	Integration Server Administrator
5	Manage the notification. See "Package Management" on page 45, "Adapter Notifications" on page 123, and "Logging and Exception Handling" on page 157 for details.	Developer and Integration Server Administrator

Ordered Notifications Transaction Processing

The following diagram and steps illustrate what happens when an Ordered Notification is invoked. The Integration Server continues to invoke the notification periodically, as defined when you configured the polling schedule parameters for the notification.

Ordered Notifications



Step	Description
1	Ordered Notifications monitor multiple insert, update, or delete operations on one or more tables by creating a single notification using the same publishable document.
2	The notification gets a connection from the service's connection pool. Adapter connections contain connection information for the database, including JDBC driver parameters.
3	The notification uses the JDBC driver to connect to the database. You created and enabled the adapter connection earlier using the Integration Server Administrator.
4	The buffer table holds the data selected by the trigger. While the adapter remains enabled, the trigger continues to monitor the database table and insert data into the buffer table. With Ordered Notifications, the adapter creates the trigger, sequence, and buffer tables for each operation you want to monitor when you enable the notification. The database trigger monitors the table(s) and inserts data into the buffer table. When the Integration Server invokes the notification, the notification will poll all of the buffer tables and publish the data in the same order in which the operations occurred. This ensures that the order of the operations is preserved.
5	The notification retrieves the rows of data from the buffer table. Each Ordered Notification generates one row for each operation. The notification uses the Operation ID and an Operation Type field you specified when you configured the notification to uniquely identify this row. The Operation ID is user-defined.
6	The notification creates the publishable document, which contains a row of data, including the Operation ID and Operation Type, from the buffer table. The notification publishes the publishable document.

Step	Description
7	Using an Integration Server trigger you configured to use the notification's publishable document, a flow or Java service on the Integration Server is invoked to react to the data changes contained in the publishable document. The flow service that processes the publishable document for the Ordered Notification needs to check the Operation ID field in the document and retrieve data from the record with the name identified by the Operation ID for processing. For example, a flow service checks to see if the Operation ID has a value of UPDATE. If this is true, then the flow service picks up the data from the UPDATE record as input and processes it. If the Operation ID value is INSERT, the flow service picks up data from the INSERT record as input and processes accordingly. See "Configuring an Integration Server Trigger and Flow Service" on page 36 for more information about using triggers and flow services with Ordered Notifications. After the data is published, the data is deleted from the buffer table.

Polling Notification Support in Clusters

The JDBC Adapter provides the ability to enable multiple instances of the same polling notification in your Integration Server clusters, and to coordinate their schedules and execution.

For more information, see "JDBC Adapter Polling Notification Support in Clusters" on page 50.

Polling Notifications and States

Before version 6.5, the Integration Server provided two states in which polling notifications could exist: Enabled and Disabled. If users wanted to stop the activities of a polling notification—for example, to perform system maintenance or make minor changes to the polling notification's configuration (such as changing its schedule)—they were forced to disable the polling notification. However, disabling a polling notification could result in data loss.

Beginning with version 6.5, the Integration Server added a third state: Suspended. The Suspended state provides a mechanism for temporarily stopping the activities of a polling notification without data loss.

The table below summarizes the states and how they affect the triggers, buffer tables, and data processing of a polling notification.

State name	Status of trigger and buffer table when polling notification enters this state	Data processing while in this state	Comments
Enabled	Database trigger and buffer table are created.	The polling notification performs as scheduled.	
Suspended	Database trigger and buffer table persist. Table retains its rows.	The polling notification is removed from the scheduler and does not execute while suspended. Any instances executing at the time the Suspended state is initiated are unaffected.	The Suspended state is available only in Integration Server version 6.5. You may suspend polling notifications in an Enabled state. You may not suspend polling notifications in a Disabled state. You may copy or export suspended polling notifications. You may not move, rename, or delete suspended polling notifications.
Disabled	Database trigger and buffer table are dropped.	The polling notification is removed from the scheduler and does not execute.	

The table above applies to Insert Notifications, Update Notifications, Delete Notifications, and Ordered Notifications. However, the table above does not apply to Basic Notifications or Stored Procedure Notifications because with these, the resource administrator (not the JDBC Adapter) is responsible for maintaining the trigger and buffer table.

For instructions on enabling, suspending, and disabling polling notifications, see the explanation of the **State** field on page 151.

webMethods Manager Support for the JDBC Adapter

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

- Adapter connections
- Adapter services
- Polling notifications

See the *webMethods Manager Server Programmer's Guide* for information about the object interfaces available to the JDBC Adapter. These object interfaces are the specific attributes and operations that you can perform to manage the JDBC Adapter components.

Viewing Different Perspectives of the webMethods Developer

Beginning with version 6.1, webMethods Developer 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.

Viewing the Adapter's Update Level

When using the JDBC Adapter with Integration Server 6.5, you can view the list of updates that have been applied to the adapter. The list of updates appears in the Updates field on the adapter's About page in the Integration Server Administrator.

Note: Updates that were created before the release of Integration Server 6.5 will not appear in the Updates field.

webMethods

CHAPTER 2

Package Management

Overview

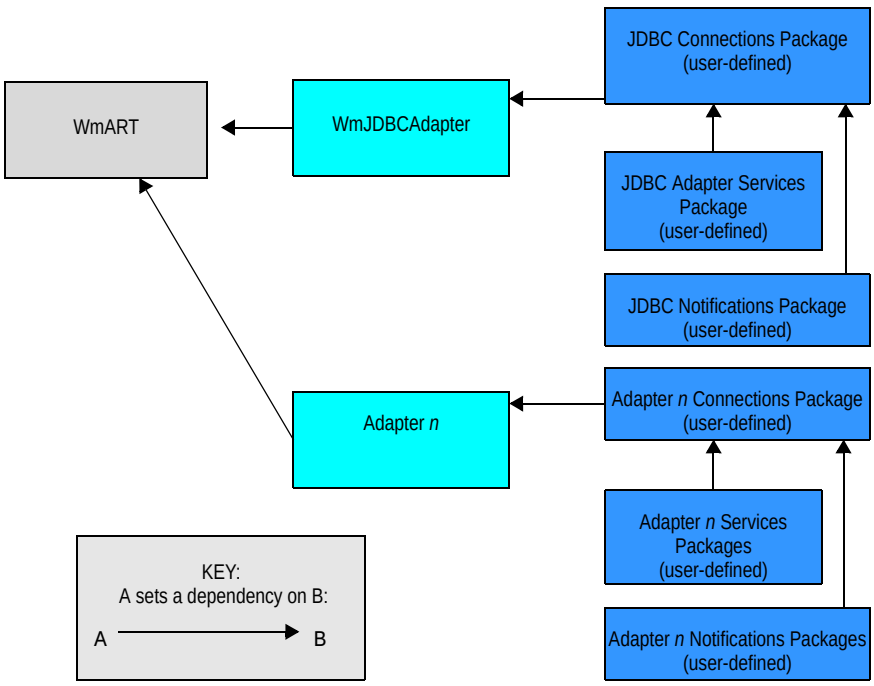
The following sections describe how to set up and manage your JDBC Adapter packages, set up Access Control Lists (ACL), and use the adapter in a clustered environment.

JDBC Adapter Package Management

The JDBC Adapter is provided as a package called WmJDBCAdapter. You manage the WmJDBCAdapter package as you would manage any package on the Integration Server.

When you create connections, adapter services, and adapter notifications, define them in user-defined packages rather than in the WmJDBCAdapter 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 WmJDBCAdapter package. That way, when the WmJDBCAdapter 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 47)
- "Enabling and Disabling Packages" on page 48
- "Importing and Exporting Packages" on page 49
- "Group Access Control" on page 49

Package Dependency Requirements and Guidelines

This section contains a list of 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, WmJDBCAdapter. (The WmJDBCAdapter package has a dependency on the WmART package.)
- Package dependencies ensure that 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 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 have a dependency on the adapter package.
  - Packages that contain adapter services must have a dependency 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 48.

- 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. For information about disabling packages, see "Enabling and Disabling Packages" on page 48.
- You can name connections, adapter services, and notifications the same name provided that they are in different folders and packages.

Enabling and Disabling Packages

All packages are automatically enabled by default. When you want to temporarily prohibit access to the elements in a package, disable the package. When you disable a package, the server unloads all of its elements from memory. Disabling a package prevents the Integration Server from loading that package at startup.

To enable a package

1 Open the Integration Server Administrator if it is not already open.

2 In the Packages menu of the navigation area, click Management.

3 Click No in the Enabled column. The server displays a  and Yes in the Enabled column.

Note: Enabling an adapter package will *not* cause its associated user-defined package(s) to be reloaded. For information about reloading packages, see the *webMethods Developer User's Guide*.

Important! Before you manually enable a user-defined package, you must first enable its associated adapter package (WmJDBCAdapter). 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.

To disable a package

1 Open the Integration Server Administrator if it is not already open.

2 In the Packages menu of the navigation area, click Management.

3 Click Yes in the Enabled column for the package that you want to disable. The server issues a prompt to verify that you want to disable the package. Click OK to enable the package. When the package is disabled, the server displays No in the Enabled column.

A disabled adapter will:

- Remain disabled until you explicitly enable it using the Administrator.
- Not be listed in the webMethods Developer.

Importing and Exporting Packages

You import and export packages using the Developer. Exporting allows you to export the package to a .zip file and save it to your hard drive. The .zip file can then be imported for use by another package.

Important! Do not rename packages you export; the rename function is comparable to moving a package, and when you import the renamed package, you lose any triggers, connections, and notifications associated with this package.

For details about importing and exporting packages, see the *webMethods Developer User's Guide*.

Group Access Control

To control which groups have access to which adapter services, use access control lists (ACLs). For example, 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 information about assigning and managing ACLs, see the *webMethods Developer User's Guide*.

JDBC Adapter in a Clustered Environment

What is webMethods Integration Server Clustering?

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

The clustering feature uses a shared *cluster store* to hold webMethods 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: webMethods 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 running webMethods Integration Server to the cluster.

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

JDBC Adapter Polling Notification Support in Clusters

This section contains these subsections:

Polling Notification Support in Clusters with Integration Server Version 6.0.1 SP2

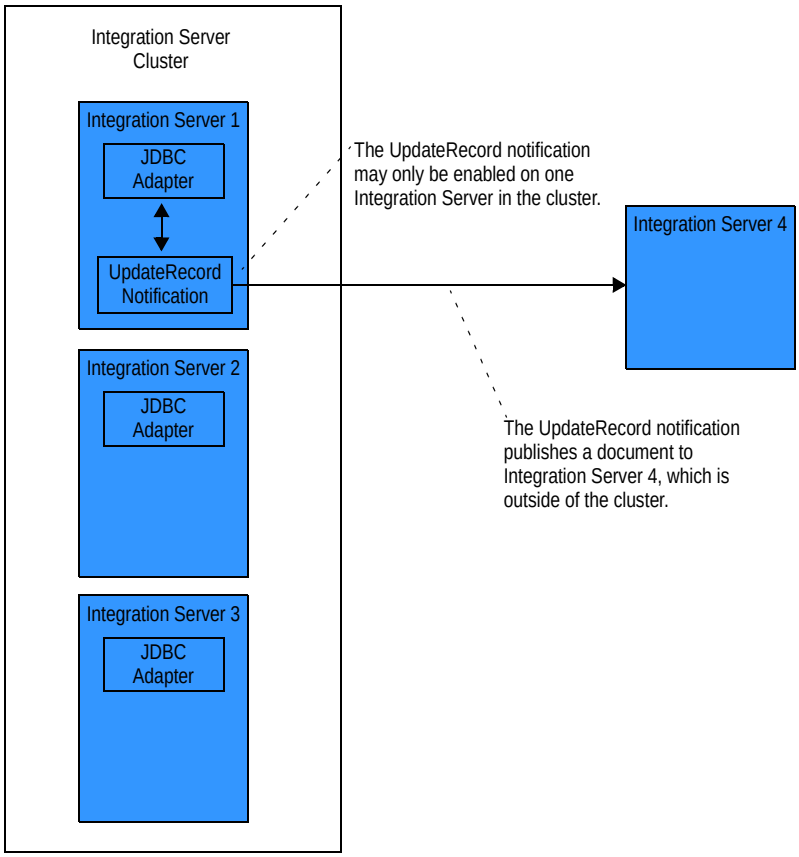
The Integration Server and the JDBC Adapter do not support having more than a single instance of a given polling notification enabled within a cluster.

When using the JDBC Adapter's polling notifications in a clustered environment, you may only enable a single instance of a given notification on one Integration Server within the cluster. If you run multiple instances of a given notification within the cluster, you will encounter document ordering problems, and the adapter will produce error messages. Additionally, even when you run a single instance of a notification within a cluster, you could encounter document duplication problems. These issues and limitations are described in the following sections.

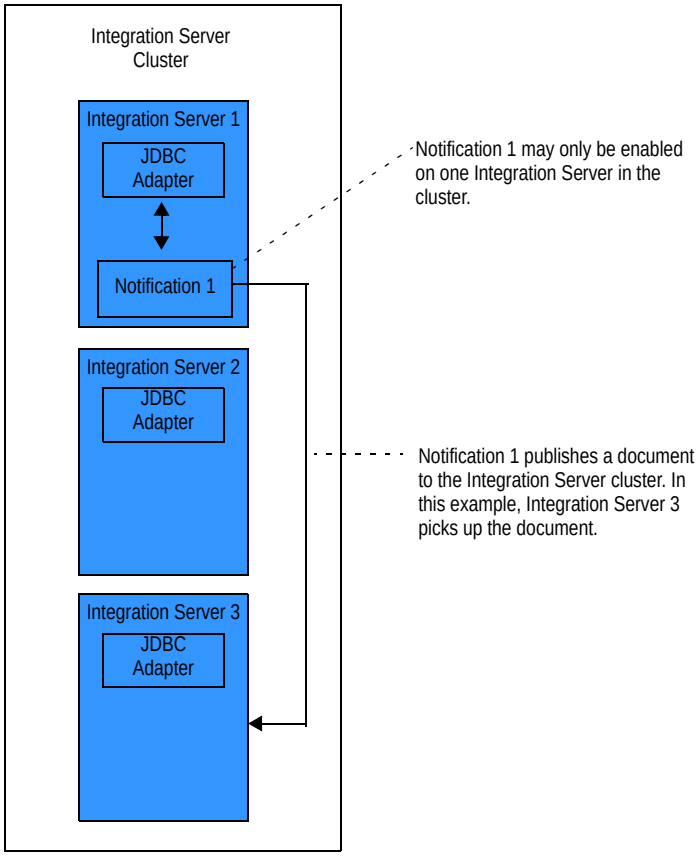
Duplicate Documents

When using the JDBC Adapter with versions of the Integration Server earlier than 6.1, the Integration Server provides duplicate document detection using its client-side queuing facility. However, when an Integration Server is added to a cluster, the server's client-side queuing mechanism becomes disabled, preventing the Integration Server from detecting duplicate documents. To work around this issue when using versions of the Integration

Server earlier than 6.1, you can enable a single instance of a given notification within the cluster and then publish the notification's documents to an Integration Server that is not included in the cluster, as shown in the following diagram.



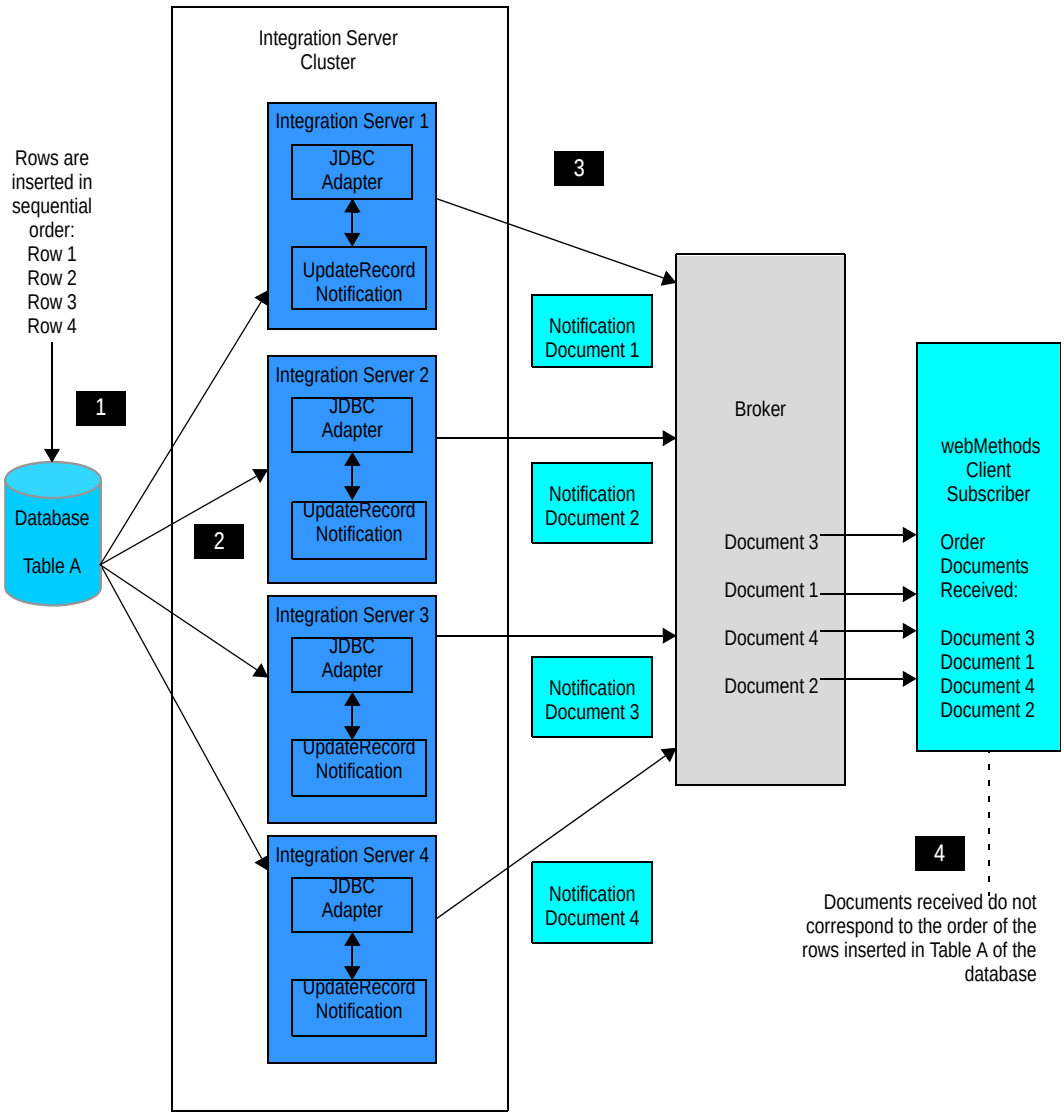
When using the JDBC Adapter with Integration Server 6.1 or later, the server provides support for duplicate document detection in a cluster using its publication and subscribe facility. See the *Publish-Subscribe Developer's Guide* for details about configuring this facility. Using this mechanism, you can enable a single instance of a given polling notification within a cluster of Integration Servers and not receive duplicate documents, as shown in the following diagram.



Ordering of Documents

If you enable more than one instance of a polling notification within a cluster, the Integration Servers in the cluster cannot maintain the order in which the documents were originally obtained from the database when the Integration Servers publish those documents. Because the Integration Server does not coordinate the order in which a message is published between servers, messages can be received in a different order than the order in which they were read from the database. The following diagram illustrates why the Integration Server cannot maintain message order when there are multiple instances of a polling notification enabled in a cluster.

Suppose you have a cluster of four Integration Servers. Each server has the JDBC Adapter installed, and all are connected to the same Broker, which has a single client. All servers in the cluster are running the same UpdateNotification (*UpdateRecord* in this example) and pulling rows from the same table (*Table A*).


Integration Server Cluster
Integration Server 1
JDBC Adapter
UpdateRecord Notification
Integration Server 2
JDBC Adapter
UpdateRecord Notification
Integration Server 3
JDBC Adapter
UpdateRecord Notification
Integration Server 4
JDBC Adapter
UpdateRecord Notification
Rows are inserted in sequential order:
Row 1
Row 2
Row 3
Row 4
1
Database
Table A
2
3
Notification Document 1
Notification Document 2
Notification Document 3
Notification Document 4
Broker
Document 3
Document 1
Document 4
Document 2
webMethods Client Subscriber
Order Documents Received:
Document 3
Document 1
Document 4
Document 2
4
Documents received do not correspond to the order of the rows inserted in Table A of the database

Step	Description
1	Rows are inserted into the database in sequential order.
2	The notifications are polling in random order and read the rows in the following order: ■ Integration Server 1: Update Notification reads row 1 ■ Integration Server 2: Update Notification reads row 2 ■ Integration Server 3: Update Notification reads row 3 ■ Integration Server 4: Update Notification reads row 4
3	Because the Integration Servers cannot coordinate the order in which the documents were received, the servers publish the notification documents in the following order: ■ Integration Server 1: Notification Document 3 ■ Integration Server 3: Notification Document 1 ■ Integration Server 2: Notification Document 4 ■ Integration Server 4: Notification Document 2
4	The client subscriber receives the documents in the following order, which is the order in which they were published by the Broker: ■ Notification Document 3 ■ Notification Document 1 ■ Notification Document 4 ■ Notification Document 2 Therefore the documents received do not correspond to the order of the rows inserted in Table A of the database.

Polling Notification Support in Clusters with Integration Server Version 6.5

When using the JDBC Adapter with Integration Server 6.5, the JDBC Adapter enables the coordinated execution of polling notifications within a webMethods Integration Server cluster. The JDBC Adapter provides the ability to enable multiple instances of the same polling notification in your cluster, and to coordinate their schedules and execution. This provides enhanced quality of service by allowing configurations for automated failover between notifications and distributed processing of polling notifications.

You can configure a polling notification to run in either Standby or Distributed mode. You can also configure additional settings for clustered polling notifications.

Standby Mode and Distributed Mode

In Standby mode, a particular instance of a polling notification will execute the notification according to its configured schedule. When you start the cluster, the polling notification that executes the first scheduled run is considered to be the primary notification. This instance will continue to execute the scheduled runs as long as it is enabled and fully functional. If at any time this notification becomes disabled, another notification in the cluster will assume control. The notification that assumes control is arbitrary. After a notification has control, it will continue to execute the schedule for as long as it is enabled and fully functional.

In Distributed mode, *any* instance of the polling notification can execute the currently scheduled run. The notification that executes the current scheduled run is arbitrary. If a notification does not complete executing within the amount of time specified in the Max Process Time field, the system considers that notification to be "dead." (For details about Max Process Time, see "Notification-Specific Settings" on page 59.) Another (enabled) instance in the cluster will recognize this situation and will attempt to execute the scheduled run.

Settings

All settings that pertain to clustered polling notifications are ignored or disabled until you include the server in a cluster. All settings have default values. There are three levels of settings: server-wide, adapter-specific, and notification-specific.

- Server-Wide Settings

  The server-wide settings are common to all webMethods 6.x adapters running on the Integration Server. The server uses them in an algorithm that determines whether a polling notification instance should be considered "dead."

Server-wide Setting Name	Values and Description
watt.art.clusteredPollingNotification.keepAliveInterval	An integer value specifying the frequency (in milliseconds) at which a notification instance tells the cluster it is still alive. Default: The value of maxlockduration in the repository2.cnf file.
watt.art.clusteredPollingNotification.keepAliveExpireTimeout	An integer value specifying the number of milliseconds that a keepAliveInterval setting can be late before it is assumed that an instance has failed. Default: The value of keepAliveInterval. Note: You should allow for "clock drift." For details, see "Clock Synchronization" on page 60.

- Adapter-Specific Settings

  The adapter-specific settings apply to all the polling notifications in your JDBC Adapter.

  The JDBC Adapter generates a configuration file for your polling notification templates. This file, WmJDBCAdapter\config\clusterProperties.cnf, is an XML file that contains a pair of settings (callbackScheme and runtimeModeLimit) for each polling notification template defined in your adapter. For example, if you created notifications using the notification templates BasicNotification, UpdateNotification, and InsertNotification, you will see these settings for each template. For an example file, see "Example JDBC clusterProperties.cnf File" on page 57.

  The callbackScheme setting specifies the run-time operations the adapter should handle for the notifications, such as enabling and disabling the notifications.

  The runtimeModeLimit setting specifies the scheduling mode for the notifications: Standby mode, Distributed mode, or neither (that is, non-clustered). These modes are described in "Standby Mode and Distributed Mode" on page 55. The value you assign to this setting determines which mode(s) that the adapter users may select in the Coordination Mode field on the Polling Notification Schedule page (see "Notification-Specific Settings" on page 59).

The available values for these settings are as follows:

Adapter-specific Setting Name	Values and Description
callbackScheme	■ 0: No callback coordination. ■ 1: Default. Coordinates the enable and disable operations. ■ 2: Coordinates the startup and shutdown operations. ■ 3: Coordinates the enable, disable, startup, and shutdown operations. **Important!** This value must *always* be 1 for the JDBC Adapter.
runtimeModeLimit	■ disable: Prevents the adapter user from selecting Standby or Distribute modes on the adapter's Polling Notification Schedule page. ■ standby: Default. Enables the adapter user to select either the Disable mode or the Standby mode. ■ distribute: Enables the adapter user to select either the Disable, Standby or Distribute mode. For more information, see "Standby Mode and Distributed Mode" on page 55.

Example JDBC clusterProperties.cnf File

The following example clusterProperties.cnf file for the JDBC Adapter shows the entries for all the possible templates associated with the JDBC polling notifications. This example file enables all notifications to be configured with a Coordination Mode of Distributed.

```
<?xml version="1.0"?>
<clusterProps>
  <pollingNotifications>
    <!--
      The following entries define the default settings used when the
      ART framework adds a new template entry to the file. For the JDBC
      Adapter the callbackSchema must always be 1.
    -->
    <callbackScheme>1</callbackScheme>
    <runtimeModeLimit>distribute</runtimeModeLimit>

     <!-- The template specific settings used for any JDBC DeleteNotification-->
    <template className="com.wm.adapter.wmjdbc.notifications.DeleteNotification">
      <callbackScheme>1</callbackScheme>
      <runtimeModeLimit>distribute</runtimeModeLimit>
    </template>

     <!--The template specific settings for JDBC StoredProcedureNotification-->
    <template className="com.wm.adapter.wmjdbc.notifications.ProcedureNotification">
      <callbackScheme>1</callbackScheme>
      <runtimeModeLimit>distribute</runtimeModeLimit>
    </template>

     <!-- The template specific settings used for any JDBC BasicNotification -->
    <template className="com.wm.adapter.wmjdbc.notifications.BasicNotification">
      <callbackScheme>1</callbackScheme>
      <runtimeModeLimit>distribute</runtimeModeLimit>
    </template>

    <!-- The template specific settings used for any JDBC UpdateNotification -->
   <template className="com.wm.adapter.wmjdbc.notifications.UpdateNotification">
      <callbackScheme>1</callbackScheme>
      <runtimeModeLimit>distribute</runtimeModeLimit>
    </template>

    <!-- The template specific settings used for any JDBC InsertNotification -->
    <template className="com.wm.adapter.wmjdbc.notifications.InsertNotification">
      <callbackScheme>1</callbackScheme>
      <runtimeModeLimit>distribute</runtimeModeLimit>
    </template>

    <!-- The template specific settings used for any JDBC OrderedNotification-->
    <template className="com.wm.adapter.wmjdbc.notifications.OrderedNotification">
      <callbackScheme>1</callbackScheme>
      <runtimeModeLimit>distribute</runtimeModeLimit>
    </template>
  </pollingNotifications>
</clusterProps>
```

- Notification-Specific Settings

  The notification-specific settings enable you to configure certain scheduling aspects of polling notifications on an individual basis.

  Two new fields appear on the Polling Notification Schedule page: Coordination Mode and Max Process Time. These fields become editable when you add your Integration Server to a cluster.

  - The Coordination Mode field controls the coordination of the notification schedules across the cluster. Depending on the value you assigned to the runtimeModeLimit setting (see "Adapter-Specific Settings" on page 56), the adapter user will be able to select some combination of the following values in the Coordination Mode field as follows:

This runtimeModeLimit value ...	Displays these values in the Coordination Mode field ...
disable	disable
standby	disable and standby
distribute	disable, standby, and distribute

  - The Max Process Time field enables other notifications to determine whether a currently executing notification should be considered to be "dead." If a notification executes a scheduled run and it fails to complete before the Max Process Time, then another notification instance will consider it dead; this other notification will assume control and execute a scheduled run. The default value is equal to the value in the watt.art.clusteredPollingNotification.keepAliveExpireTimeout setting in the server.cnf file.

    If the Max Process Time setting is not high enough, you may encounter a situation in which a notification is executing normally but another notification assumes it is "dead." When the original notification completes, it will recognize that it was prematurely considered "dead." In this case, the system logs an Illegal Overlap exception with message id [ART.116.3715]. If this exception occurs, increase your Max Process Time setting.

    When setting the value of Max Process Time, you should allow for "clock drift." For details, see "Clock Synchronization" on page 60.

  If you want to update the schedule and settings of a notification in a cluster, all notification instances in the cluster must be suspended or disabled for the changes to be saved. If any notification instance in the cluster is enabled, the adapter will not save the updates.

  If all instances of a notification in the cluster do not have the same settings, the notification that became active first will have precedence.

Clock Synchronization

To determine whether a notification has failed, notifications use the system clocks of the machines that host the clustered Integration Servers. Synchronizing the clocks of all machines in the cluster is critical for the proper execution of clustered polling notifications.

However, in time these clocks might become un-synchronized. Therefore you should anticipate the effect of "clock drift" when you establish values for the keepAliveExpireTimeout and Max Process Time settings. Clock drift is the time difference between the clocks. As a guideline, add to the keepAliveExpireTimeout and Max Process Time settings two times the maximum clock drift you anticipate.

Configuring JDBC Polling Notifications

To enable JDBC polling notifications to operate in Distributed mode

1. In the cluster, shut down the Integration Server you are configuring.
2. Open the WmJDBCAdapter\config\clusterProperties.cnf file.
3. Change all `<runtimeModeLimit>` values to `<distribute>`. This enables all templates to operate in Distributed mode. For an example clusterProperties.cnf file, see "Example JDBC clusterProperties.cnf File" on page 57. Alternatively, you may update only specific template entries.
4. Save the file and restart the Integration Server.
5. Start the Integration Server Administrator.
6. From the Adapters menu in the navigation area of the Administrator, select JDBC Adapter.
7. From the navigation area, select Polling Notifications.
8. From the JDBC Adapter Polling Notifications table, use the Enabled field to disable the notification, and then click the Edit Schedule icon.
9. Set the Coordination Mode to Distributed.
10. Use the Enabled field to enable the notification.

Important! To maintain duplicate detection and ordering, your polling notification schedules must *not* run with the Overlap option selected. (To access the Overlap option, click the Edit Schedule icon.)

After you configure a polling notification, you may propagate all the affected components across your cluster. Changing the polling notification schedule from the Integration Server Administrator or changing the polling notification settings in the webMethods Developer will require you to propagate the polling notification across the cluster. If you made

changes to the settings in server.cnf or to the WmJDBCAdapter\config\clusterProperties.cnf files, you must also propagate these across the cluster.

Adapter Service Support in Clusters

Adapter services are supported in a clustered environment. In order for a cluster to handle requests identically, you should be sure the identical service is in each server in the cluster so that if a given service is not available, the request can be redirected and handled by another server in the cluster.

See "Clustering Considerations and Requirements" on page 62 for more details about adapter services in clusters.

Replicating Packages to webMethods Integration Servers

Every webMethods Integration Server in the cluster should contain an identical set of packages that you define using the JDBC Adapter; that is, you should replicate the JDBC 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 49, a server that cannot handle a client's service request can automatically redirect the request to another server in the cluster. However, the JDBC 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 create two JDBC 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 webMethods 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 JDBC 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	All Integration Servers in the cluster must be the same version, with the same service packs and fixes applied.
Adapter packages	All adapter packages on one Integration Server should be replicated to all other Integration Servers in the cluster.

All Integration Servers in a given cluster must have identical...	For Example...
Adapter connections	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. If you plan to use connection pools in a clustered environment, see "Considerations When Configuring Connections with Connection Pooling Enabled" on page 63.
Adapter services	If you configure a specific InsertSQL 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. 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.

See "Replicating Packages to webMethods Integration Servers" on page 61 for information about replicating adapter packages, connections, and adapter services across multiple Integration Servers in a cluster.

Considerations When Installing JDBC Adapter Packages

For each Integration Server in the cluster, use the standard JDBC Adapter installation procedures for each machine, as described in the *webMethods JDBC 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 JDBC Adapter, see "Configuring JDBC Adapter Connections" on page 68.

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

webMethods

CHAPTER 3

JDBC Adapter Connections

Overview

This chapter describes how to configure and manage JDBC Adapter connections. For more information about how adapter connections work, see "Adapter Connections" on page 13.

Before Configuring or Managing Adapter Connections

To prepare to configure or manage adapter connections

1. Install the webMethods Integration Server and the JDBC Adapter on the same machine. See the *webMethods JDBC Adapter Installation Guide* for details.
2. Install a compatible JDBC driver. See "Installing the JDBC Driver on the Integration Server" on page 67 for instructions. See the *webMethods JDBC Adapter Installation Guide* for a list of supported drivers.
3. Make sure you have webMethods administrator privileges so that you can access the JDBC Adapter's administrative screens. See the *webMethods Integration Server Administrator's Guide* for information about setting user privileges.
4. Be sure to see "Database Driver Known Limitations" on page 187 for a list of known driver limitations since it may affect how you configure your connections.
5. Start your Integration Server and the Administrator, if they are not already running.
6. Using the Administrator, make sure the WmJDBCAdapter package is enabled. See "Enabling and Disabling Packages" on page 48 for instructions.
7. 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 "JDBC Adapter Package Management" on page 46.
8. If you use Oracle JDBC OCI drivers, you must set an environment variable before you can configure the connection. See "Setting the Environment Variable for Oracle JDBC OCI Drivers" on page 67.

Setting the Environment Variable for Oracle JDBC OCI Drivers

For Oracle JDBC OCI drivers, you must set the following environment variable before you configure the connection. Also check that the OCI client is configured correctly before you proceed.

Platform	Environment Variable Setting
Solaris*	`LD_LIBRARY_PATH=/ORACLE_HOME/lib`
HP*	`SHLIB_PATH=/ORACLE_HOME/lib`
AIX*	`LIBPATH=/ORACLE_HOME/lib`
Linux	`LD_LIBRARY_PATH=/ORACLE_HOME/lib`

Note: *If you are using Oracle 920 JDBC driver files with an Oracle 920 client to connect to different Oracle database versions, set the environment variable for your platform to `/ORACLE_HOME/lib32`.

Installing the JDBC Driver on the Integration Server

You must install the JDBC driver on the Integration Server before you can specify connections. The Integration Server requires access to the Java classes for each JDBC driver that it will use to connect to a database. You must place the Java classes in a location that the server can access. Typically, you place the Java classes in the server's classpath.

To place the classes in the server's classpath, place the .zip or .jar file containing the classes in the IntegrationServer/package/WmJDBCAdapter/code/jars directory. Restart the Integration Server. The server will automatically add the .zip or .jar libraries to its classpath after you restart.

See the *webMethods JDBC Adapter Installation Guide* for a list of supported drivers.

Configuring JDBC Adapter Connections

When you configure JDBC Adapter connections, you specify information that the Integration Server uses to connect to a JDBC system. You configure JDBC Adapter connections using the Administrator.

Note: If you will be using Oracle JDBC OCI drivers with the JDBC Adapter, you must add an environment variable setting before you configure adapter connections. See "Setting the Environment Variable for Oracle JDBC OCI Drivers" on page 67 for details.

To configure a connection

1 In the Adapters menu in the Administrator's navigation area, click JDBC Adapter.

2 On the Connections screen, click Configure New Connection.

3 On the Connection Types screen, click JDBC Adapter Connection to display the Configure Connection Type screen.

4 In the JDBC Adapter Connection section, use the following fields.

Field	Description/Action
Package	The package in which to create the connection. You must create the package using the Developer before you can specify it using this parameter. For general information about creating packages, see the *webMethods Developer User's Guide*. Note: Configure the connection in a user-defined package rather than in the adapter's package. See "JDBC Adapter Package Management" on page 46 for other important considerations when creating packages for the JDBC Adapter.
Folder Name	The folder in which to create 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 *webMethods Developer User's Guide* for more information about the use of special characters in package, folder, and element names.

5 In the Connection Properties section, use the following fields:

Note: The following table shows suggested values for these parameters as guidance only. See your JDBC driver documentation for more information about what values to assign these parameters.

a Specify the Transaction Type and DataSource Class Name as follows:

Field	Description/Action
Transaction Type	The type of transaction support that the connection provides. Select one of the following transaction types: NO_TRANSACTION: The connection automatically commits operations. LOCAL_TRANSACTION: The connection uses local transactions. If you plan to use the connection with BatchInsertSQL or BatchUpdateSQL adapter services, you must specify LOCAL_TRANSACTION type. Note: If you are configuring a Basic Notification and using the Exactly Once Notification and Delete selected records options, you must configure the notification to use a LOCAL_TRANSACTION connection. See step 11d on page 141 for information about these specific configuration options. XA_TRANSACTION: The connection uses XA transactions. Note: Do not use XA_TRANSACTION connections with Teradata databases. Note: The JDBC Adapter does support the Oracle RAC TAF facility (which provides failover support) for Oracle v.9.2.x using an OCI driver. Under these circumstances you must use LOCAL_TRANSACTION connections. For a more detailed description of the transaction support provided by the JDBC Adapter, see "Transaction Management of JDBC Adapter Connections" on page 15.

Field	Description/Action
DataSource Class Name	The name of the JDBC driver's DataSource or XADataSource class. Type one of the following DataSource or XADataSource class names, depending on the JDBC driver and transaction type that the connection will use.

	Oracle JDBC OCI Driver
	
	For NO_TRANSACTION and LOCAL_TRANSACTION:
	
	`oracle.jdbc.pool.OracleDataSource`
	
	For XA_TRANSACTION:
	
	`oracle.jdbc.xa.client.OracleXADataSource`

	Oracle JDBC Thin Driver
	
	For NO_TRANSACTION and LOCAL_TRANSACTION:
	
	`oracle.jdbc.pool.OracleDataSource`
	
	For XA_TRANSACTION:
	
	`oracle.jdbc.xa.client.OracleXADataSource`

	DataDirect Connect for JDBC, edition 3.2 and 3.3 (for Microsoft SQL Server) For all transaction types: `com.ddtek.jdbcx.sqlserver.SQLServerDataSource`
	Microsoft SQL Server 2000 Driver for JDBC 2.2.0019 For all transaction types: `com.microsoft.jdbcx.sqlserver.SQLServerDataSource`
	Microsoft SQL Server 2005 Driver for JDBC 1.0.809.102 For all transaction types: `com.microsoft.sqlserver.jdbc.SQLServerDataSource`

	JTOpen v4.1 (for DB2 for AS/400 v4r5, v5r1, and v5r2)
	
	For NO_TRANSACTION and LOCAL_TRANSACTION:
	
	`com.ibm.as400.access.AS400JDBCDataSource`
	
	For XA_TRANSACTION:
	
	`com.ibm.as400.access.AS400JDBCXADataSource`

Field	Description/Action
DataSource Class Name (continued)	DB2 net type 3 (for OS/390 v6 and v7, and UDB 7.2 and 8.1) For NO_TRANSACTION and LOCAL_TRANSACTION: `COM.ibm.db2.jdbc.DB2DataSource`
	DataDirect Connect for JDBC 3.2 (for UDB 7.2) For NO_TRANSACTION and LOCAL_TRANSACTION: `com.ddtek.jdbcx.db2.DB2DataSource`
	DataDirect Connect for JDBC 3.2 (for UDB 8.1) For all transaction types: `com.ddtek.jdbcx.db2.DB2DataSource`

	DB2 app type 2 (for UDB 7.2 and 8.1)
	For NO_TRANSACTION and LOCAL_TRANSACTION:
	`COM.ibm.db2.jdbc.DB2DataSource`
	For XA_TRANSACTION:
	`COM.ibm.db2.jdbc.DB2XADataSource`

	DB2 Universal type 4 For NO_TRANSACTION and LOCAL_TRANSACTION: com.ibm.db2.jcc.DB2SimpleDataSource

	jCONNECT 5.5 type 4 (For Sybase v. 11.x and 12.x)
	For NO_TRANSACTION and LOCAL_TRANSACTION:
	`com.sybase.jdbc2.jdbc.SybDataSource`
	For XA_TRANSACTION:
	`com.sybase.jdbc2.jdbc.SybXADataSource`

Field	Description/Action

DataSource Class Name (continued)	Informix JDBC 2.21 type 4 (For Informix 7.31 and 9.x)
	For NO_TRANSACTION and LOCAL_TRANSACTION:
	`com.informix.jdbcx.IfxDataSource`
	Note: If you use the com.informix.jdbcx.IfxDataSource DataSource class with Integration Server version 6.0.1 SP2, you must disable the WmTomcat package. Be aware that disabling the WmTomcat package also disables support for any JSPs. For general information about setting dependencies, see "JDBC Adapter Package Management" on page 46. For more detailed information, see the *webMethods Developer User's Guide*.
	For XA_TRANSACTION:
	`com.informix.jdbcx.IfxXADataSource`

	Teradata type 4 (for Teradata v2r5) For NO_TRANSACTION and LOCAL_TRANSACTION: `com.ncr.teradata.TeraDataSource`

b Depending on the driver type, some or all of the following fields are required. To determine whether your driver type requires you to use a field, see "Required Connection Property Fields" on page 80.

Note: If you use a DataDirect Connect for JDBC driver you must create the package and port information you enter from this tab. See your DataDirect Connect documentation for details.

Field	Description/Action
serverName	The name of the server that hosts the database.
user*	The user name that the connection will use to connect to the database.
password*	The password for the database user name specified in user.
Retype password	Retype the password you just entered.
databaseName	The database to which the connection will connect.

Field	Description/Action
portNumber	The port number that the connection will use to connect to the database. Note: The DB2 net type 3 driver property portNumber is the same as the DB2 JDBC Applet server's port number. The default is 6789.
networkProtocol	A standard JDBC DataSource property to indicate the name of the network protocol that the connection will use when connecting to the database. Use this field only if you use an Oracle JDBC OCI driver. Type `tcp`, which is the name of the network protocol.

Note: *For Microsoft SQL Server databases: The username and password you configure for a connection must be the same as those used to create the tables you use with a specific notification; otherwise, an exception will be generated at run time.

6 There are two types of property settings you can specify in the Other Properties field:

- Table filter property settings: limit which catalogs, schemas and tables to list. See step a below.
- Driver-dependent property settings: enable you to provide additional JDBC driver DataSource properties. See step b below.

Use a ; (semi-colon) to delimit table filter and driver-dependent property settings. Do not enter spaces after the semi-colon. For example:

```
TableFilter='<current catalog>'.'Accounting';driverType=oci
```

a Table Filter Property Settings limit the list of catalogs, schemas, and tables you select when you create adapter services and notifications. This setting is beneficial if you work with large databases.

- Use the following format to enter Table Filter Property Settings in the Other Properties field:

```
TableFilter='catalog1'.'schema1'.'table1',
'catalog2'.'schema2'.'table2','catalogN'.'schemaN'
```

Example:

```
TableFilter='Payables'.'Accounting'.'Finance'
```

Note: The TableFilter setting is case-sensitive. Be sure that the names you enter match the names in the database table. If you use `'<current catalog>'` or `'<current schema>'` described below, be sure that you use all lowercase letters.

- When configuring the TableFilter property:
  - You can use `'<current catalog>'` as the *`catalog`* for the default login catalog.
  - You can use `'<current schema>'` as the *`schema`* for the login user.
  - The *`table`* variable is the table name pattern and is optional. If you do not specify a *`table`* value, the JDBC Adapter loads all of the tables for the schema. The following example lists all the tables under the Accounting schema:

    ```
    TableFilter='<current catalog>'.'Accounting'
    ```

Note: Informix databases automatically access the current catalog only.

  - You can use the exact table name or any of the following special characters in a String:
  - Use % to match any substring of zero or more characters. The following example lists all the tables under the Accounting schema named Finance, Finance1, FinanceDept, and so forth:

    ```
    TableFilter='<current catalog>'.'Accounting'. 'Finance%'
    ```

  - Use _ to match any one character. The following example lists all the tables under the Accounting schema named Finance1, Finance2, Finance3, and so forth.

    ```
    TableFilter='<current catalog>'.'Accounting'. 'Finance_'
    ```

  - Use , (commas) to list multiple TableFilter settings. Do not enter spaces after the comma. For example:

    ```
    TableFilter='<current catalog>'.'Accounting'.
    'Finance_','<current catalog>'.'Employee%'
    ```

  - Use ; (semi-colons) to delimit multiple property settings; that is, both TableFilter settings and driver-dependent settings. Do not enter spaces after the semi-colon. For example:

    ```
    TableFilter='<current catalog>'.'Accounting'.
    'Finance_','<current catalog>'.'Accounting';driverType=thin
    ```

b You also use the Other Properties field to set driver-dependant property settings, which enable you to provide additional JDBC driver DataSource properties depending on the driver you use.

- Use the following format:

  propertyName=value

- Use ; (semi-colons) to delimit multiple property settings; that is, both TableFilter settings and driver-dependent settings. Do not enter spaces after the semi-colon. For example:

  ```
  TableFilter='<current catalog>'.'Accounting'.
  'Finance';selectMethod=cursor
  ```

In the Other Properties field, type the following suggested driver-dependent parameters based on the JDBC driver and the transaction type the connection is using:

Field	Description/Action

Other Properties (Driver-dependent Properties)	DataDirect Connect for JDBC 3.2 for DB2 UDB 7.2 and 8.1:
	`PackageName=value`
	where *value* is the name of the package you created earlier in the database. See your DataDirect Connect for JDBC documentation for information about creating packages.
	DB2 Universal type 4:
	`driverType=4`
	Required setting for this driver. If the `driverType` is not set to 4, then Type 2 connectivity will be selected by default.
	`readOnly=true`
	Creates a read only connection.
	`currentSchema=YourSchemaName`
	Specifies the default schema name used to qualify unqualified database objects in dynamically prepared SQL statements.
	`loginTimeout=number`
	The maximum time in seconds to wait for the DataSource object to connect to a data source.

Field	Description/Action

Other Properties (continued)	`traceFile=`*`fileName`*
	Specifies the name of a file into which this driver writes trace information.
	`traceFileAppend=true`
	Appends, instead of overwrites, the file that is specified by the traceFile property.
	`traceLevel=`*`number`*
	Specifies what level to trace. This property's data is INTEGER. Set *`number`* to `-1` to TRACE_ALL or `2` to TRACE_STATEMENT_CALLS. See your vendor's driver documentation for more information.
	DataDirect Connect for JDBC edition 3.2 and 3.3 for Microsoft SQL Server:
	For XA_TRANSACTION only:
	`selectMethod=cursor`
	Microsoft SQL Server 2000 Driver for JDBC:
	For XA_TRANSACTION only:
	`selectMethod=cursor`
	Microsoft SQL Server 2005 Driver for JDBC:
	For XA_TRANSACTION only:
	`selectMethod=cursor`
	Oracle JDBC Thin Driver:
	`driverType=thin`
	Oracle JDBC OCI Driver version 8i:
	`driverType=oci8`
	Oracle JDBC OCI Driver version 9i:
	`driverType=oci`

Field	Description/Action

	Teradata Type 4:
	`DSName=value`
	where *value* is the Teradata database server name.
	Informix JDBC 2.21 type 4 (for NO_TRANSACTION. LOCAL_TRANSACTION, and XA_TRANSACTION):
	`IfxIFXHOST=hostname`
	where *hostname* is the machine name of the database server.

7 In the Connection Management Properties section, use the following fields:

Field	Description/Action
Enable Connection Pooling	Enables the connection to use connection pooling. See "Adapter Connections" on page 13 for more information about connection pooling. 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 63.
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.

Field	Description/Action
Block Timeout	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. 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. 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.
Expire Timeout	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. 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. 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. 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.

Field	Description/Action
Startup Retry Count	(This field applies only to Integration Server versions 6.1 and later.) 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	(This field applies only to Integration Server versions 6.1 and later.) The number of seconds that the system should wait between attempts to initialize the connection pool.

8 Click Save Connection.

If the parameters for the connection are valid, the Integration Server enables the connection. Otherwise, the Integration Server does not enable the connection.

The connection you created appears on the adapter's Connections screen and in the Developer Service Browser.

Required Connection Property Fields

The following table lists the required Connection Properties tab fields by driver type. For details about the following fields, see step 5b on page 72.

Driver Name	serverName	user	password	databaseName	portNumber	Network Protocol
DB2 net type 3 (for OS/390 V6 and V7, and UDB V7.2 and V8.1)	Yes	Yes	Yes	Yes	Yes	No
DB2 Universal type 4	Yes	Yes	Yes	Yes	Yes	No
Teradata Type 4	No	Yes	Yes	No	No	No
DataDirect Connect for JDBC 3.2 (for UDB 7.2 and 8.1)	Yes	Yes	Yes	Yes	Yes	No
DB2 app type 2 (for UDB V7.2 and V8.1)	No	Yes	Yes	Yes	No	No
Microsoft SQL Server 2000 (for JDBC 2.20019)	Yes	Yes	Yes	No	Yes	No
Microsoft SQL Server 2005 (for JDBC 1.0.809.102)	Yes	Yes	Yes	No	Yes	No
Microsoft SQL Server 7 with DataDirect Connect for JDBC	Yes	Yes	Yes	No	Yes	No
Oracle Thin	Yes	Yes	Yes	Yes	Yes	No
Oracle OCI	Yes	Yes	Yes	Yes	Yes	Yes
JTOpen v4.1 (for DB2 for AS/400 v4r5, v5r1, and v5r2)	Yes	Yes	Yes	No	No	No
Informix JDBC 2.21 type 4 (Informix v. 7.31 and 9.x)	Yes	Yes	Yes	Yes	Yes	No
jCONNECT 5.5 type 4 (for Sybase v. 11.x and 12x)	Yes	Yes	Yes	Yes	Yes	No

Dynamically Changing a Service's Connection at Runtime

Beginning with Integration Server 6.5, you may run a service using a connection other than the default connection that was associated with the service when the service was created. To override the default, you must code your flow to pass a value through the pipeline into a service's $connectionName field.

For example, you have a flow whose primary purpose is to update a production database. However, you want the flow to have the capability to update a test database – with the decision of which database to update to be made programmatically at runtime. The output signature of the flow's first service contains a field called Target. The flow could branch based on the value in Target. If Target contains the value Production, the second service in the flow would ignore $connectionName – thus using its default connection to connect to (and then update) the production database. However, if Target contains the value Test, the second service in the flow would use the value in the $connectionName from the pipeline and connect to (and then update) the test database.

Keep in mind these restrictions when using dynamic connections:

- Both connections—the default and override—must use the same database schema.
- The connection with which you override the default (that is, the value provided for $connectionName) must be configured to use the same transaction type as the default connection.
- The $connectionName field is present only in services…
  - created with Developer 6.5
  - created with Developer 6.1 (or earlier) and edited with Developer 6.5

For more information, see "Changing the Connection Associated with an Adapter Service at Runtime" on page 20.

Viewing Adapter Connection Parameters

You can view a connection's parameters from the 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 JDBC Adapter.

2 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 JDBC Adapter Connections" on page 68.

3 Click Return to JDBC Adapter Connections to return to the main connections screen.

To view the parameters for a connection using the Developer

1 Start the Developer if it is not already running.

2 From the Developer navigation area, open the package and folder in which the connection is located.

3 Click the connection you want to view.

The parameters for the connection appear on the Connection Information tab. For descriptions of the connection parameters, see "Configuring JDBC Adapter Connections" on page 68.

Editing Adapter Connections

If the login information for a database changes, or if you want to redefine parameters that a connection uses when connecting to a database, you can update a connection's parameters using the Administrator.

To edit a connection

1 In the Adapters menu in the Administrator's navigation area, click JDBC Adapter.

2 Make sure that the connection is disabled before editing it. See "Disabling Adapter Connections" on page 85 for instructions.

3 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 JDBC Adapter Connections" on page 68.

4 Click **Save Changes** to save the connection and return to the Connections screen.

Copying Adapter Connections

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

To copy a connection

1 In the **Adapters** menu in the Administrator's navigation area, click **JDBC Adapter**.

2 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 JDBC Adapter Connections" on page 68.

3 Click **Save Connection** to save the connection and return to the Connections screen.

Deleting Adapter Connections

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

If you delete a JDBC Adapter connection, the adapter services or notifications that are defined to use the connection will no longer work. However, depending on which version of the Integration Server you use, you may be able to assign a different connection to an adapter service and re-use the service, as follows:

- When using versions of the Integration Server earlier than 6.1, 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 82 for instructions.) Therefore, if you delete a JDBC Adapter connection, any adapter services or notifications that are defined to use the connection will no longer work.
- When using Integration Server 6.1 or later, you *can* change which connection an adapter service uses. Therefore, if you delete a JDBC Adapter connection, you can assign a different connection to an adapter service and re-use the service. To do this, you use the built-in webMethods function setAdapterServiceNodeConnection. For more information, see "Changing the Connection Associated with an Adapter Service or Notification at Design Time" on page 19.

To delete a connection

1. In the Adapters menu in the Administrator navigation area, click JDBC Adapter.
2. Make sure that the connection is disabled before deleting. To disable the connection, click Yes in the Enabled column and click OK to confirm. The Enabled column now shows No (Disabled) for the connection.
3. On the Connections screen, click  for the connection you want to delete.

   The Integration Server deletes the adapter connection.

Enabling Adapter Connections

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

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 navigation area, click JDBC 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 a  and Yes in the Enabled column.

Disabling Adapter Connections

JDBC Adapter connections must be disabled before you can edit or delete them. You disable adapter connections using the Administrator.

To disable a connection

1 In the Adapters menu in the Administrator navigation area, click JDBC 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.

webMethods

CHAPTER 4

Adapter Services

Overview

This chapter describes how to configure and manage JDBC Adapter services. For detailed descriptions of the available JDBC Adapter services, see "Adapter Services" on page 17.

Before Configuring or Managing Adapter Services

To prepare to configure or manage JDBC Adapter services

1. Start your Integration Server and the Administrator, if they are not already running.
2. Make sure you have Integration Server Administrator privileges so that you can access the JDBC Adapter's administrative screens. See the *webMethods Integration Server Administrator's Guide* for information about setting user privileges.
3. Be sure to check "Database Driver Known Limitations" on page 187 for a list of known limitations for your database driver since it may affect how you configure your connections and adapter services.
4. If you have made changes to the table schema for a given adapter service, be sure to update the adapter service accordingly.
5. Using the Administrator, make sure the WmJDBCAdapter package is enabled. See "Enabling and Disabling Packages" on page 48 for instructions.
6. Using the Administrator, configure an adapter connection to use with the adapter service. See "Configuring JDBC Adapter Connections" on page 68 for instructions.

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

Beginning with version 6.1, the Integration Server solves this limitation by providing a built-in service you can use at design time to change the connection associated with an adapter service. For more information, see "Changing the Connection Associated with an Adapter Service or Notification at Design Time" on page 19.

7. Start the Developer if it is not already running.
8. If you are using the Developer 6.1 or later, make sure you are viewing the webMethods Developer in the Edit perspective, as described in "Viewing Different Perspectives of the webMethods Developer" on page 43.
9. 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 WmJDBCAdapter package. For

more information about managing packages for the adapter, see Chapter 2, "Package Management" on page 45.

Configuring SelectSQL Services

A SelectSQL service retrieves specified information from a database table. You configure JDBC adapter services using the Developer. For more information about adapter services, see "Using Adapter Services" on page 19.

Be sure to review the section "Before Configuring or Managing Adapter Services" on page 88 before you configure adapter services.

To configure a SelectSQL service

1 From the Developer File menu, select New...

2 Select Adapter Service from the list of elements and click Next.

3 Select JDBC Adapter as the adapter type and click Next.

4 Select the appropriate Adapter Connection Name and click Next.

5 Select the SelectSQL template and click Next.

6 Type a unique name for the service and select the appropriate folder. Click Finish.

The adapter service editor for the adapter service appears. 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.

7 Select the Tables tab to configure the database table (or tables) the operation accesses, using the following fields:

Field	Description/Action
Table Alias	The table alias is assigned automatically when you select more than one table in the Table Name field. The default is `t1`.
Table Name	Select a table name. The default for the associated catalog name is `current catalog`. The default for the associated schema name is `current schema`. The table name must not contain a period. If the table name does contain a period, Developer will throw a Java exception. Note: Informix databases do not allow you to specify a catalog and database name because you can only access the current catalog.

Field	Description/Action
Type	The type displays automatically based on the table you select.

8 If you are not joining tables, skip this step. Select the Joins tab to specify the columns for joining the tables you just configured.

a Select the Insert Row icon to create new left and right columns.

b Select Left Column and select the first table's joining column.

c Select the appropriate Operator.

d Select Right Column and select the next table's joining column.

e Repeat this procedure until you have defined all the joins.

9 Use the SELECT tab to define the columns and fields to be selected as follows:

a In the ALL/DISTINCT field, select ALL to include duplicate rows or DISTINCT to suppress duplicate rows. Selecting ALL corresponds to the SQL statement `SELECT ALL name from tablename`. The default value is blank, which corresponds to the SQL statement `SELECT name from tablename`.

b Use the Insert Row icon (or the Fill in all rows to the table icon) to create new rows as needed.

c As you insert additional rows, the corresponding Column Type, JDBC Type, Output Field Type, and Output Field display for each column you select in the Expression field.

Note: You can use the BigDecimal data type with the JDBC Adapter. However, the webMethods Developer does not support the BigDecimal data type. This means that the Developer will not correctly display a BigDecimal data type result and you cannot enter a value of this data type. However, the JDBC Adapter will process the BigDecimal data type correctly.

Use the following fields:

Field	Description/Action
Expression	The column name in the database table.
Column Type	The column data type defined in the database table.
JDBC Type	The JDBC type of the corresponding Output Field.

Field	Description/Action
Output Field Type	The data type of the output field. The JDBC Adapter automatically converts database-specific types to Java data types.
Output Field	The name of the field containing the output from the SELECT operation. An output field name displays when you select an expression. You can also modify the output field names as required.
Sort Order	Specifies how rows are returned as follows: Select either Ascend or Descend. Leave the field blank if there is no sort order.
Maximum Row	Use this field only to specify the maximum number of records to retrieve from the database. The default value of `0` (no limit) retrieves all records.

10 Use the WHERE tab to specify the conditions for selecting information:

a Select the Insert Row icon  to define new WHERE clause fields.

b Select a logical operator from the AND/OR field, an Operator, and separators (the left and right parentheses) as needed, and use the following fields:

Field	Description/Action
AND/OR	The logical operator.
Column	The name of the column you want to use in the WHERE clause.
Operator	The operator used with the Column and Input Field.
Input Field*	The default value is ?, which acts as a placeholder for the variable so that you can set the input variable for that column at run time, or get input external to this adapter service. You can also type a fixed value in this field now or at run time. If you type a fixed value, be sure that it is valid, or an exception will be generated at run time.
JDBC Type	The JDBC type of the corresponding Input Field.
Input Field Type	The corresponding input field's Java type. For a list of JDBC type to Java type mappings, see "JDBC Data Type to Java Data Type Mappings" on page 170.

Field	Description/Action
Input Field (second occurrence of this field)	Type the name of the input field. If you use the default ? variable placeholder as the Input Field value in the WHERE clause, be sure to enter the corresponding Input Field and its JDBC Field Type in the same order as they appear on the top portion of the WHERE tab.

Note: * For Oracle users, if you use a CHAR data type and enter a value in the Input Field, you must pad the field with spaces equal to the total column width. For example, if you use the value `Smith` in the input field and define it as a CHAR data type of 50 bytes, you must pad the field with 45 spaces. An alternative is to use a DynamicSQL adapter service because you do not need to pad this field when you use a SQL statement. Use the `${value}` to replace the ? variable placeholder, such as `WHERE char_col = ${value}`. This value will be in the input list so that at run time you can set the value to a String without having to pad the column width. See "Configuring DynamicSQL Services" on page 111 for an example and configuration instructions.

c If necessary, use the Shift Up or Shift Down icons to change the order of the WHERE clause to ensure the parameters are parsed in the correct order.

d Repeat this procedure until you have specified all WHERE parameters.

11 For information about using the Input/Output, Settings, Audit, Permissions, and Results tabs, see the *webMethods Developer User's Guide*. These tabs apply to all services that you configure using versions of the Developer earlier than 6.1.

When using Developer 6.1 or later, the Developer only contains the Adapter Settings and Input/Output tabs. The information from the Audit and Permissions tabs appears in the Properties panel, and the information from the Results tab appears in the Results panel.

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

To run or test the service directly, see "Testing Adapter Services" on page 119.

Configuring InsertSQL Services

An InsertSQL service inserts new information into a database table. You configure JDBC adapter services using the Developer. For more information about adapter services, see "Using Adapter Services" on page 19.

Be sure to review the section "Before Configuring or Managing Adapter Services" on page 88 before you configure adapter services.

To configure an InsertSQL service

1 From the Developer File menu, select New...

2 Select Adapter Service from the list of elements and click Next.

3 Select JDBC Adapter as the adapter type and click Next.

4 Select the appropriate Adapter Connection Name and click Next.

5 Select the InsertSQL template and click Next.

6 Type a unique name for the service and select the appropriate folder. Click Finish.

The adapter service editor for the adapter service appears. 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.

7 Select the Table tab to configure the database table to be updated and set the fields as follows:

Field	Description/Action
Table Name	Select a table name. The default for the associated catalog name is `current catalog`. The default for the associated schema name is `current schema`. The table name must not contain a period. If the table name does contain a period, Developer will throw a Java exception. Note: Informix databases do not allow you to specify a catalog and database name because you can only access the current catalog.
Type	The table type displays automatically based on the table you select.

8 Select the INSERT tab and use the Column, Column Type, JDBC Type and Expression fields on the top row of the tab to define the columns and fields to be inserted as described in the following table.

a Use the Insert Row icon  (or the Fill in all rows to the table icon ) to create new rows as needed.

Field	Description/Action
Column	The INSERT column name in the database table.
Column Type	The INSERT column data type in the database table.

Field	Description/Action
JDBC Type	The JDBC type for the input field.
Expression	The default value is ?, which acts as a placeholder for the variable so that you can set the input variable for that column at run time, or get input external to this adapter service. It adds one row with the same column name to the table. You can also type a fixed value in this field now or at run time. If you type a fixed value, be sure that it is valid, or an exception will be generated at run time.

b For each inserted row that uses the default Expression value of ?, the corresponding JDBC Type, Input Field, and Input Field Type display on the second row of the INSERT tab.

Note: You can use the BigDecimal data type with the JDBC Adapter. However, the webMethods Developer does not support the BigDecimal data type. This means that the Developer will not correctly display a BigDecimal data type result and you cannot enter a value of this data type. However, the JDBC Adapter will process the BigDecimal data type correctly.

Use the following fields:

Field	Description/Action
Column	The INSERT column name in the database table.
Column Type	The INSERT column data type in the database table.
JDBC Type	The JDBC type for the input field.
Input Field*	The input field name. You can change this name if needed.
Input Field Type	The data type of the input field. You can change this type if needed.

Note: * For Oracle users, if you use a CHAR data type and enter a value in the Input Field, you must pad the field with spaces equal to the total column width. For example, if you use the value `Smith` in the input field and define it as a CHAR data type of 50 bytes, you must pad the field with 45 spaces. An alternative is to use a DynamicSQL adapter service because you do not need to pad this field when you use a SQL statement. Use the `${value}` to replace the `?` variable placeholder, such as `WHERE char_col = ${value}`. This value will be in the input list so that at run time you can set the value to a String without having to pad the column width. See "Configuring DynamicSQL Services" on page 111 for an example and configuration instructions.

9 Use the Result tab's Result Field and Result Field Type to specify the output field name and corresponding field types for the resulting number of rows that have been inserted.

10 For information about using the Input/Output, Settings, Audit, Permissions, and Results tabs, see the *webMethods Developer User's Guide*. These tabs apply to all services that you configure using versions of the Developer earlier than 6.1.

When using Developer 6.1 or later, the Developer only contains the Adapter Settings and Input/Output tabs. The information from the Audit and Permissions tabs appears in the Properties panel, and the information from the Results tab appears in the Results panel.

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

To run or test the service directly, see "Testing Adapter Services" on page 119.

Configuring UpdateSQL Services

An UpdateSQL service updates existing information in a database table and includes a mapping for an output field that stores the number of rows affected by the update operation. You configure JDBC adapter services using the Developer. For more information about adapter services, see "Using Adapter Services" on page 19.

Be sure to review the section "Before Configuring or Managing Adapter Services" on page 88 before you configure adapter services.

To configure an UpdateSQL service

1 From the Developer File menu, select New...

2 Select Adapter Service from the list of elements and click Next.

3 Select JDBC Adapter as the adapter type and click Next.

4 Select the appropriate Adapter Connection Name and click Next.

5 Select the UpdateSQL template and click Next.

6 Type a unique name for the service and select the appropriate folder. Click Finish.

The adapter service editor for the adapter service appears. 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.

7 Select the Table tab to configure the database table to be updated and set fields as follows:

Field	Description/Action
Table Name	Select a table name. The default for the associated catalog name is `current catalog`. The default for the associated schema name is `current schema`. The table name must not contain a period. If the table name does contain a period, Developer will throw a Java exception. Note: Informix databases do not allow you to specify a catalog and database name because you can only access the current catalog.
Type	The table type displays automatically based on the table you select.

8 Select the UPDATE tab and use the Column, Column Type, JDBC Type and Expression fields on the top row of the tab to define the columns and fields, as follows:

a Use the Insert Row icon (or the Fill in all rows to the table icon) to create new rows as needed.

Field	Description/Action
Column	The UPDATE column name in the database table.
Column Type	The UPDATE column data type in the database table.
JDBC Type	The JDBC type of the corresponding Input Field.
Expression	The default value is ?, which acts as a placeholder for the variable so that you can set the input variable for that column at run time, or get input external to this adapter service. It adds one row with the same column name to the table. You can also type a fixed value in this field now or at run time. If you type a fixed value, be sure that it is valid, or an exception will be generated at run time.

b If you insert additional rows using the default Expression value of ?, the corresponding JDBC Type, Input Field and Input Field Type display on the second row of the UPDATE tab:

Field	Description/Action
Column	The UPDATE column name in the database table.
Column Type	The column data type defined in the database table.
JDBC Type	The JDBC type of the input field.
Input Field*	The input field name. You can change this name if needed.
Input Field Type	The data type of the input field. You can change this type if needed.

Note: * For Oracle users, if you use a CHAR data type and enter a value in the Input Field, you must pad the field with spaces equal to the total column width. For example, if you use the value `Smith` in the input field and define it as a CHAR data type of 50 bytes, you must pad the field with 45 spaces. An alternative is to use a DynamicSQL adapter service because you do not need to pad this field when you use a SQL statement. Use the `${value}` to replace the `?` variable placeholder, such as `WHERE char_col = ${value}`. This value will be in the input list so that at run time you can set the value to a String without having to pad the column width. See "Configuring DynamicSQL Services" on page 111 for an example and configuration instructions.

9 Use the WHERE tab to specify the conditions for selecting information:

a Select the Insert Row icon  to define new WHERE clause fields.

b Select a logical operator from the AND/OR field, an Operator, and separators (the left and right parentheses) as needed.

Note: You can use the BigDecimal data type with the JDBC Adapter. However, the webMethods Developer does not support the BigDecimal data type. This means that the Developer will not correctly display a BigDecimal data type result and you cannot enter a value of this data type. However, the JDBC Adapter will process the BigDecimal data type correctly.

Use the following fields:

Field	Description/Action
AND/OR	The logical operator.
Column	The name of the column you want to use in the WHERE clause.
Operator	The operator used with the Column and Input Field.
Input Field*	The default value is ?, which acts as a placeholder for the variable so that you can set the input variable for that column at run time, or get input external to this adapter service. You can also type a fixed value in this field now or at run time. If you type a fixed value, be sure that it is valid, or an exception will be generated at run time.
JDBC Type	The JDBC type of the corresponding Input Field.

Note: * For Oracle users, if you use a CHAR data type and enter a value in the Input Field, you must pad the field with spaces equal to the total column width. For example, if you use the value `Smith` in the input field and define it as a CHAR data type of 50 bytes, you must pad the field with 45 spaces. An alternative is to use a DynamicSQL adapter service because you do not need to pad this field when you use a SQL statement. Use the `${value}` to replace the `?` variable placeholder, such as `WHERE char_col = ${value}`. This value will be in the input list so that at run time you can set the value to a String without having to pad the column width. See "Configuring DynamicSQL Services" on page 111 for an example and configuration instructions.

10 Use the Result tab's Result Field and Result Field Type to specify the output field name and corresponding field types for the resulting number of rows that have been inserted.

11 For information about using the Input/Output, Settings, Audit, Permissions, and Results tabs, see the *webMethods Developer User's Guide*. These tabs apply to all services that you configure using versions of the Developer earlier than 6.1.

When using Developer 6.1 or later, the Developer only contains the Adapter Settings and Input/Output tabs. The information from the Audit and Permissions tabs appears in the Properties panel, and the information from the Results tab appears in the Results panel.

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

To run or test the service directly, see "Testing Adapter Services" on page 119.

Configuring BatchInsertSQL Services

Similar to an InsertSQL service, a BatchInsertSQL service also inserts new information into a database table; however the BatchInsertSQL service can insert a large volume of data into a table more efficiently than an InsertSQL service, improving performance when a large data volume is involved. You configure JDBC adapter services using the Developer. For more information about adapter services, see "Using Adapter Services" on page 19.

Be sure to review the section "Before Configuring or Managing Adapter Services" on page 88 before you configure adapter services.

Note: BatchInsertSQL services do not work with Teradata databases.

To configure a BatchInsertSQL Service

1 From the Developer **File** menu, select **New...**

2 Select **Adapter Service** from the list of elements and click **Next**.

3 Select **JDBC Adapter** as the adapter type and click **Next**.

4 Select the appropriate **Adapter Connection Name** and click **Next**.

Note: For BatchInsertSQL services, you must use a LOCAL_TRANSACTION connection. If you do not use LOCAL_TRANSACTION, you will not see a list of tables in the **Tables** tab. Also, you may not see an error message until you reload metadata values or check the error log. See "Configuring JDBC Adapter Connections" on page 68 for instructions for creating a LOCAL_TRANSACTION connection. See "Reloading Adapter Values" on page 121 for information about reloading metadata values.

5 Select the **BatchInsertSQL** template and click **Next**.

6 Type a unique name for the service and select the appropriate folder. Click **Finish**.

The adapter service editor for the adapter service appears. 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.

7 Select the Table tab to configure the database table to be updated and set the fields as follows:

Field	Description/Action
Table Name	Select a table name. The default for the associated catalog name is `current catalog`. The default for the associated schema name is `current schema`. The table name must not contain a period. If the table name does contain a period, Developer will throw a Java exception. Note: Informix databases do not allow you to specify a catalog and database name because you can only access the current catalog.
Type	The table type displays automatically based on the table you select.

8 Select the INSERT tab and use the Column, Column Type, JDBC Type and Expression fields on the top row of the tab to define the columns and fields to be inserted as described in the following table.

a Use the Insert Row icon (or the Fill in all rows to the table icon) to create new rows as needed.

Field	Description/Action
Column	The INSERT column name in the database table.
Column Type	The INSERT column data type in the database table.
JDBC Type	The JDBC type for the input field.
Expression	The default value is `?`, which acts as a placeholder for the variable so that you can set the input variable for that column at run time, or get input external to this adapter service. It adds one row with the same column name to the table. You can also type a fixed value in this field now or at run time. If you type a fixed value, be sure that it is valid, or an exception will be generated at run time.

b For each inserted row that uses the default **Expression** value of `?`, the corresponding **Input Field**, and **Input Field Type** display on the second row of the INSERT tab. Use the following fields:

Note: You can use the BigDecimal data type with the JDBC Adapter. However, the webMethods Developer does not support the BigDecimal data type. This means that the Developer will not correctly display a BigDecimal data type result and you cannot enter a value of this data type. However, the JDBC Adapter will process the BigDecimal data type correctly.

Field	Description/Action
Column	The INSERT column name in the database table.
Column Type	The INSERT column data type in the database table.
Input Field*	The input field name. You can change this name if needed.
Input Field Type	The data type of the input field. You can change this type if needed. **Note:** If you use WmFlatFile services to generate the document list as input, the input field type must be java.lang.String. This is because fields from WmFlatFile services generate documents that have String fields.

Note: * For Oracle users, if you use a CHAR data type and enter a value in the **Input Field**, you must pad the field with spaces equal to the total column width. For example, if you use the value `Smith` in the input field and define it as a CHAR data type of 50 bytes, you must pad the field with 45 spaces. An alternative is to use a DynamicSQL adapter service because you do not need to pad this field when you use a SQL statement. Use the `${value}` to replace the `?` variable placeholder, such as `WHERE char_col = ${value}`. This value will be in the input list so that at run time you can set the value to a String without having to pad the column width. See "Configuring DynamicSQL Services" on page 111 for an example and configuration instructions.

9 Use the **Batch Result** tab's **Batch Result Output Name** to specify the output field name for the batch operation. The output of the batch operation is a string list. The elements of the string list are ordered according to the order in which commands were added to

the batch. Depending on the JDBC driver you use, the elements in the string list may be one of the following:

- A number greater than or equal to zero. This indicates that the command was successfully executed and the number of rows in the database affected.
- A value of SUCCESS_NO_INFO. This indicates that the command was processed successfully but the number of rows affected is unknown.

10 For information about using the Input/Output, Settings, Audit, Permissions, and Results tabs, see the *webMethods Developer User's Guide*. These tabs apply to all services that you configure using versions of the Developer earlier than 6.1.

When using Developer 6.1 or later, the Developer only contains the Adapter Settings and Input/Output tabs. The information from the Audit and Permissions tabs appears in the Properties panel, and the information from the Results tab appears in the Results panel.

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

To run or test the service directly, see "Testing Adapter Services" on page 119.

Configuring BatchUpdateSQL Services

Similar to an UpdateSQL service, a BatchUpdateSQL service updates information in a database table; however, the BatchUpdateSQL service can update a large volume of data in a table more efficiently than an UpdateSQL service, improving performance when a large data volume is involved. You configure JDBC adapter services using the Developer. For more information about adapter services, see "Using Adapter Services" on page 19.

Note: BatchUpdateSQL services do not work with Teradata databases.

Be sure to review the section "Before Configuring or Managing Adapter Services" on page 88 before you configure adapter services.

To configure a BatchUpdateSQL Service

1 From the Developer File menu, select New...

2 Select Adapter Service from the list of elements and click Next.

3 Select JDBC Adapter as the adapter type and click Next.

4 Select the appropriate Adapter Connection Name and click Next.

Note: For BatchUpdateSQL services, you must use a LOCAL_TRANSACTION connection. If you do not use LOCAL_TRANSACTION, you will not see a list of tables in the Tables tab. Also, you may not see an error message until you reload metadata values or check the error log. See "Configuring JDBC Adapter Connections" on page 68 for instructions for creating a LOCAL_TRANSACTION connection. See "Reloading Adapter Values" on page 121 for information about reloading metadata values.

5 Select the BatchUpdateSQL template and click Next.

6 Type a unique name for the service and select the appropriate folder. Click Finish.

The adapter service editor for the adapter service appears. 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.

7 Select the Table tab to configure the database table to be updated and set the fields as follows:

Field	Description/Action
Table Name	Select a table name. The default for the associated catalog name is `current catalog`. The default for the associated schema name is `current schema`. The table name must not contain a period. If the table name does contain a period, Developer will throw a Java exception. Note: Informix databases do not allow you to specify a catalog and database name because you can only access the current catalog.
Type	The table type displays automatically based on the table you select.

8 Select the UPDATE tab and use the Column, Column Type, JDBC Type, and Expression fields on the top row of the tab to define the columns and fields, as follows:

a Use the Insert Row icon (or the Fill in all rows to the table icon) to create new rows as needed.

Field	Description/Action
Column	The UPDATE column name in the database table.
Column Type	The UPDATE column data type in the database table.

Field	Description/Action
JDBC Type	The JDBC type of the corresponding Input Field.
Expression	The default value is ?, which acts as a placeholder for the variable so that you can set the input variable for that column at run time, or get input external to this adapter service. It adds one row with the same column name to the table. You can also type a fixed value in this field now or at run time. If you type a fixed value, be sure that it is valid, or an exception will be generated at run time.

b If you insert additional rows using the default Expression value of ?, the corresponding Input Field and Input Field Type display on the second row of the UPDATE tab:

Note: You can use the BigDecimal data type with the JDBC Adapter. However, the webMethods Developer does not support the BigDecimal data type. This means that the Developer will not correctly display a BigDecimal data type result and you cannot enter a value of this data type. However, the JDBC Adapter will process the BigDecimal data type correctly.

Field	Description/Action
Column	The UPDATE column name in the database table.
Column Type	The column data type defined in the database table.
Input Field*	The input field name. You can change this name if needed.
Input Field Type	The data type of the input field. You can change this type if needed. Note: If you use WmFlatFile services to generate the document list as input, the input field type must be java.lang.String. This is because fields from WmFlatFile services generate documents are have String fields.

Note: * For Oracle users, if you use a CHAR data type and enter a value in the Input Field, you must pad the field with spaces equal to the total column width. For example, if you use the value `Smith` in the input field and define it as a CHAR data type of 50 bytes, you must pad the field with 45 spaces. An alternative is to use a DynamicSQL adapter service because you do not need to pad this field when you use a SQL statement. Use the `${value}` to replace the `?` variable placeholder, such as `WHERE char_col = ${value}`. This value will be in the input list so that at run time you can set the value to a String without having to pad the column width. See "Configuring DynamicSQL Services" on page 111 for an example and configuration instructions.

9 Use the WHERE tab to specify the conditions for selecting information:

a Select the Insert Row icon  to define new WHERE clause fields.

b Select a logical operator from the AND/OR field, an Operator, and separators (the left and right parentheses) as needed. Use the following fields:

Field	Description/Action
AND/OR	The logical operator.
Column	The name of the column you want to use in the WHERE clause.
Operator	The operator used with the Column and Input Field.
Input Field	The default value is ?, which acts as a placeholder for the variable so that you can set the input variable for that column at run time, or get input external to this adapter service. You can also type a fixed value in this field now or at run time. If you type a fixed value, be sure that it is valid, or an exception will be generated at run time.
JDBC Type	The JDBC type of the corresponding Input Field.
Input Field Type*	The corresponding input field's Java type. For a list of JDBC type to Java type mappings, see "JDBC Data Type to Java Data Type Mappings" on page 170.
Input Field (second occurrence of this field)	Type the name of the input field. If you use the default ? variable placeholder as the Input Field value in the where clause, be sure to enter the corresponding Input Field and its JDBC Field Type in the same order as they appear on the top portion of the WHERE tab.

10 Use the Batch Result tab's Batch Result Output Name to specify the output field name for the batch operation. The output of the batch operation is a string list. The elements of the string list are ordered according to the order in which commands were added to the batch. Depending on the JDBC driver you use, the elements in the string list may be one of the following:

- A number greater than or equal to zero. This indicates that the command was successfully executed and the number of rows in the database affected.
- A value of SUCCESS_NO_INFO. This indicates that the command was processed successfully but the number of rows affected is unknown.

11 For information about using the Input/Output, Settings, Audit, Permissions, and Results tabs, see the *webMethods Developer User's Guide*. These tabs apply to all services that you configure using versions of the Developer earlier than 6.1.

When using Developer 6.1 or later, the Developer only contains the Adapter Settings and Input/Output tabs. The information from the Audit and Permissions tabs appears in the Properties panel, and the information from the Results tab appears in the Results panel.

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

To run or test the service directly, see "Testing Adapter Services" on page 119.

Configuring DeleteSQL Services

A DeleteSQL service deletes rows from a table and includes a mapping for an output field that stores the number of affected rows. You configure JDBC adapter services using the Developer. For more information about adapter services, see "Using Adapter Services" on page 19.

Be sure to review the section "Before Configuring or Managing Adapter Services" on page 88 before you configure adapter services.

To configure a DeleteSQL service

1 From the Developer File menu, select New...

2 Select Adapter Service from the list of elements and click Next.

3 Select JDBC Adapter as the adapter type and click Next.

4 Select the appropriate Adapter Connection Name and click Next.

5 Select the DeleteSQL template and click Next.

6 Type a unique name for the service and select the appropriate folder. Click Finish.

The adapter service editor for the adapter service appears. 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.

7 Select the Table tab to configure the database table to be updated and set the fields as follows:

Field	Description/Action
Table Name	Select a table name. The default for the associated catalog name is `current catalog`. The default for the associated schema name is `current schema`. The table name must not contain a period. If the table name does contain a period, Developer will throw a Java exception. Note: Informix databases do not allow you to specify a catalog and database name because you can only access the current catalog.
Type	The table type displays automatically based on the table you select.

8 Use the WHERE tab to specify the conditions for selecting information:

a Select the Insert Row icon  to define new WHERE clause fields.

b Select a logical operator from the AND/OR field, an Operator, and separators (the left and right parentheses) as needed.

Note: You can use the BigDecimal data type with the JDBC Adapter. However, the webMethods Developer does not support the BigDecimal data type. This means that the Developer will not correctly display a BigDecimal data type result and you cannot enter a value of this data type. However, the JDBC Adapter will process the BigDecimal data type correctly.

Use the following fields:

Field	Description/Action
AND/OR	The logical operator.
Column	The name of the column you want to use in the WHERE clause.
Operator	The operator used with the Column and Input Field.

Field	Description/Action
Input Field*	The default value is ?, which acts as a placeholder for the variable so that you can set the input variable for that column at run time, or get input external to this adapter service. You can also type a fixed value in this field now or at run time. If you type a fixed value, be sure that it is valid, or an exception will be generated at run time.
JDBC Type	The JDBC type of the corresponding Input Field.
Input Field Type	The corresponding input field's Java type. For a list of JDBC type to Java type mappings, see "JDBC Data Type to Java Data Type Mappings" on page 170.
Input Field (second occurrence of this field)	Type the name of the input field. If you use the default ? variable placeholder as the Input Field value in the where clause, be sure to enter the corresponding Input Field and its JDBC Field Type in the same order as they appear on the top portion of the WHERE tab.

Note: * For Oracle users, if you use a CHAR data type and enter a value in the Input Field, you must pad the field with spaces equal to the total column width. For example, if you use the value `Smith` in the input field and define it as a CHAR data type of 50 bytes, you must pad the field with 45 spaces. An alternative is to use a DynamicSQL adapter service because you do not need to pad this field when you use a SQL statement. Use the `${value}` to replace the ? variable placeholder, such as `WHERE char_col = ${value}`. This value will be in the input list so that at run time you can set the value to a String without having to pad the column width. See "Configuring DynamicSQL Services" on page 111 for an example and configuration instructions.

9 Use the Result tab's Result Field and Result Field Type to specify the output field name and corresponding field types for the resulting number of rows that have been inserted.

10 To verify input or output information for this service, use the Input/Output tab as needed.

11 For information about using the Input/Output, Settings, Audit, Permissions, and Results tabs, see the *webMethods Developer User's Guide*. These tabs apply to all services that you configure using versions of the Developer earlier than 6.1.

When using Developer 6.1 or later, the Developer only contains the Adapter Settings and Input/Output tabs. The information from the Audit and Permissions tabs appears in

the Properties panel, and the information from the Results tab appears in the Results panel.

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

To run or test the service directly, see "Testing Adapter Services" on page 119.

Configuring CustomSQL Services

A CustomSQL service defines and executes custom SQL to perform database operations. You can execute almost any SQL statement required by integrations, such as data management statements. You configure JDBC adapter services using the Developer. For more information about adapter services, see "Using Adapter Services" on page 19.

If you need to write custom SQL, you can create a service that uses customized SQL statements. This allows you the flexibility to execute almost any SQL statements required, such as data management statements and data definition statements, including insert, select, update, and delete.

Because an adapter service that uses custom SQL provides no error checking, be sure that your SQL statement works correctly. You can verify SQL statement accuracy using your vendor's SQL utility. See your vendor documentation for details.

Be sure to review the section "Before Configuring or Managing Adapter Services" on page 88 before you configure adapter services.

To create a CustomSQL service

Note: You can use a CustomSQL service to call a stored procedure only when the stored procedure does not have any OUT/INOUT or return parameters. If you need to use these parameters, use the StoredProcedure service. See "Configuring StoredProcedure Services" on page 115 for instructions.

1 From the Developer File menu, select New...

2 Select Adapter Service from the list of elements and click Next.

3 Select JDBC Adapter as the adapter type and click Next.

4 Select the appropriate Adapter Connection Name and click Next.

5 Select the CustomSQL template and click Next.

6 Type a unique name for the service and select the appropriate folder. Click Finish.

The adapter service editor for the adapter service appears. 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.

7 Select the SQL tab to specify a SQL statement and the associated input and output parameters.

Use the Insert Row icon  and set the SQL parameters as described in the following table:

Note: You can use the BigDecimal data type with the JDBC Adapter. However, the webMethods Developer does not support the BigDecimal data type. This means that the Developer will not correctly display a BigDecimal data type result and you cannot enter a value of this data type. However, the JDBC Adapter will process the BigDecimal data type correctly.

Field	Description/Action

SQL*	A SQL statement.
	
	If you need more space to type your statement, use the launch icon to the right to open a text editor window.
	
	You can type the statement directly in this field, for example:
	
	`select short_col, int_col, float_col, double_col, date_col, date_time_col, varchar_col from ADAPTER-TEST`
	
	For variable names, use the ? variable placeholder for each variable. For example:
	
	`select employee_name where StaffID = ? and Dept = ?`
	
	Note: Do not end your SQL statement with a semi-colon (;) or an exception will be generated at run time.
	
	Note: You may paste text into this field from the system clipboard. However, you may not cut or copy text from this field to the clipboard for pasting into another application.

Input JDBC Type	The JDBC type of the corresponding Input Field.
Input Field Type	The Java type that corresponds to the input JDBC type. For a list of JDBC type to Java type mappings, see "JDBC Data Type to Java Data Type Mappings" on page 170.
Input Field*	Type the name of the input field.
Output JDBC Type	The JDBC type of the corresponding Output Field.

Field	Description/Action
Output Field Type	The Java type that corresponds to the output JDBC type. For a list of JDBC type to Java type mappings, see "JDBC Data Type to Java Data Type Mappings" on page 170.
Output Field	The output field name.
Maximum Row	The maximum number of records to retrieve from the database. The default value of `0` (no limit) retrieves all records. Use this field only with SQL statements that return a result set.
Result Field	Name of the output field that contains the total number of rows affected by the SQL statement.
Result Field Type	The data type of the Result Field.

Note: *If you use the `?` variable placeholder(s) in your SQL statement, be sure to enter the corresponding Input Field and field type information in the same order as they appear in your SQL statement. For example, using the SQL statement:

```
select employee_name where StaffID = ? and Dept = ?
```

`StaffID` would be the first entry in the Input Field and `Dept` would be the second entry.

8 For information about using the Input/Output, Settings, Audit, Permissions, and Results tabs, see the *webMethods Developer User's Guide*. These tabs apply to all services that you configure using versions of the Developer earlier than 6.1.

When using Developer 6.1 or later, the Developer only contains the Adapter Settings and Input/Output tabs. The information from the Audit and Permissions tabs appears in the Properties panel, and the information from the Results tab appears in the Results panel.

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

To run or test the service directly, see "Testing Adapter Services" on page 119.

Configuring DynamicSQL Services

Creating a DynamicSQL service allows you to configure a dynamic SQL statement, part of which you set at run time using input fields. At run time, the service will create the SQL statement by combining the contents of the input fields and then executing it. This is useful when you need the flexibility to set all or part of a SQL statement at run time, instead of at design time.

Using Input and Output Parameters

You must specify the input and output parameters of the DynamicSQL service at design time. When you configure the service, the input fields you configure will contain the input for the SQL statement. The output fields you configure will contain the results from the result set. Be sure that the input and output fields correctly match those of the SQL statement. If there is any mismatch, the service will generate an exception at run time.

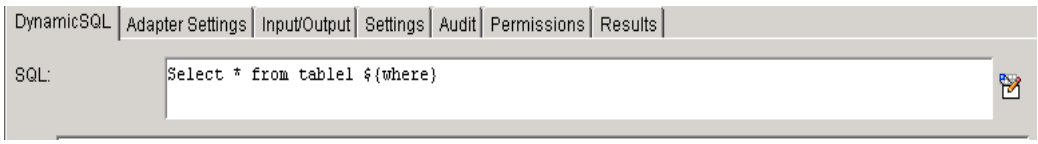
Note: You can use the BigDecimal data type with the JDBC Adapter. However, the webMethods Developer does not support the BigDecimal data type. This means that the Developer will not correctly display a BigDecimal data type result and you cannot enter a value of this data type. However, the JDBC Adapter will process the BigDecimal data type correctly.

Configuring a DynamicSQL Statement

DynamicSQL uses `${INPUT_FIELD_NAME}` to map a part of the SQL statement to the input field. At design time, the service template generates an input field with *`INPUT_FIELD_NAME`*. At run time, the service parses the statement and replaces the `${ INPUT_FIELD_NAME }` with the actual contents of the input field.

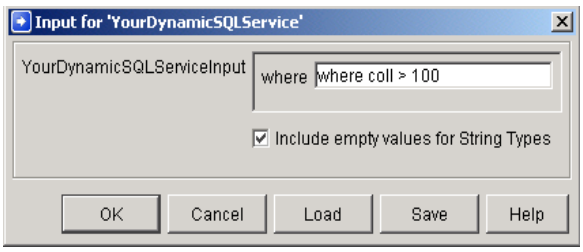
For example, consider the following DynamicSQL statement:

```
select * from table1 ${where}:
```



In this example, the service template will generate an input field for the {`where`} portion of the statement. Note that you do not type a semicolon (`;`) at the end of the SQL statement. Doing so will generate an exception at run time.

At run time, the {`where`} field is set to `where col1>100:`



The generated SQL statement will be `Select * from table1 where col1>100.`

A more extreme example would be to set the SQL field to "${sql}"; in this case, the entire SQL statement will be set through the input field `sql`.

Creating a DynamicSQL Service

Use the following instructions to create a DynamicSQL adapter service. You configure JDBC adapter services using the Developer.

Be sure to review the section "Before Configuring or Managing Adapter Services" on page 88 before you configure adapter services.

To create a DynamicSQL service

Note: You can use a DynamicSQL service to call a stored procedure only when the stored procedure does not have any OUT/INOUT or return parameters. If you need these parameters, use the StoredProcedure service. See "Configuring StoredProcedure Services" on page 115 for instructions.

1. From the Developer **File** menu, select **New...**
2. Select **Adapter Service** from the list of elements and click **Next**.
3. Select **JDBC Adapter** as the adapter type and click **Next**.
4. Select the appropriate **Adapter Connection Name** and click **Next**.
5. Select the **DynamicSQL** template and click **Next**.
6. Type a unique name for the service and select the appropriate folder. Click **Finish**.

   The adapter service editor for the adapter service appears. 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.
7. Select the **Dynamic SQL** tab to specify a SQL statement and the associated input and output parameters.

   Use the **Insert Row** icon  and set the SQL parameters as described in the table below.

Field	Description/Action

SQL*	A SQL statement.
	
	If you need more space to type your statement, use the launch icon to the right to open a text editor window.
	
	You can type the statement directly in this field, for example:
	
	`select short_col, int_col, float_col, double_col, date_col, date_time_col, varchar_col from ADAPTER-TEST`
	
	For variable names, use the `?` variable placeholder for each variable. For example:
	
	`select employee_name where StaffID = ? and Dept = ?`
	
	Note: Do not end your SQL statement with a semi-colon (`;`) or you will generate an exception.
	
	Note: You may paste text into this field from the system clipboard. However, you may not cut or copy text from this field to the clipboard for pasting into another application.

Input JDBC Type	The JDBC type of the corresponding Input Field.
Input Field Type	The Java type that corresponds to the input JDBC type. For a list of JDBC type to Java type mappings, see "JDBC Data Type to Java Data Type Mappings" on page 170.
Input Field*	Type the name of the input field.
Output JDBC Type	The JDBC type of the corresponding Output Field.
Output Field Type	The Java type that corresponds to the output JDBC type. For a list of JDBC type to Java type mappings, see "JDBC Data Type to Java Data Type Mappings" on page 170.
Output Field	The output field name.
Maximum Row	The maximum number of records to retrieve from the database. The default value of `0(no limit)` retrieves all records. Use this field only with SQL statements that return a result set.
Result Field	Name of the output field that contains the total number of rows affected by the SQL statement. Do not use `results` as the value of the Result Field.

Field	Description/Action
Result Field Type	The data type of the Result Field.

Note: *If you use the ? variable placeholder(s) in your SQL statement, be sure to enter the corresponding Input Field and field type information in the same order as they appear in your SQL statement.For example, using the SQL statement:

```
select employee_name where StaffID = ? and Dept = ?
```

`StaffID` would be the first entry in the Input Field and `Dept` would be the second entry.

8 For information about using the Input/Output, Settings, Audit, Permissions, and Results tabs, see the *webMethods Developer User's Guide*. These tabs apply to all services that you configure using versions of the Developer earlier than 6.1.

When using Developer 6.1 or later, the Developer only contains the Adapter Settings and Input/Output tabs. The information from the Audit and Permissions tabs appears in the Properties panel, and the information from the Results tab appears in the Results panel.

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

To run or test the service directly, see "Testing Adapter Services" on page 119.

Configuring StoredProcedure Services

A StoredProcedure service calls a stored procedure to perform database operations. You configure JDBC adapter services using the Developer. For more information about adapter services, see "Using Adapter Services" on page 19.

The SQL statement for an adapter service can also be a stored procedure call. A stored procedure is SQL code that is encapsulated in a statement and compiled into executable code. It is an object that is stored in the database and called when the adapter applies the SQL statement to the database.

Stored procedures provide greater flexibility in performing database operations in response to documents. You can configure operations for stored procedure calls with or without parameters. To learn how to create a stored procedure, see the vendor documentation for your database.

Be sure to review the section "Before Configuring or Managing Adapter Services" on page 88 before you configure adapter services.

To configure a StoredProcedure service

1 From the Developer File menu, select New...

2 Select Adapter Service from the list of elements and click Next.

3 Select JDBC Adapter as the adapter type and click Next.

4 Select the appropriate Adapter Connection Name and click Next.

5 Select the StoredProcedureSQL template and click Next.

6 Type a unique name for the service and select the appropriate folder. Click Finish.

The adapter service editor for the adapter service appears. 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.

7 Select the Call tab to specify the stored procedure to call. Use the following fields to set the Call parameters:

Note: You can use the BigDecimal data type with the JDBC Adapter. However, the webMethods Developer does not support the BigDecimal data type. This means that the Developer will not correctly display a BigDecimal data type result and you cannot enter a value of this data type. However, the JDBC Adapter will process the BigDecimal data type correctly.

Field	Description/Action
Catalog Name	The name of the catalog. The default for the catalog name is `current catalog`.
Schema Name	The name of the schema. The default for the schema name is `current schema`.
Enable Procedure Name Lookup (Optional)	To type in the Procedure Name, set this field to `False`. To select the Procedure Name from a list, set this field to `True`. The default is `False`. To save you time, use the default value (typing the name) if you know the name of the procedure and you are working with a large database which may have a long list of procedures.
Procedure Name	Type or select the stored procedure name, depending on how you set the Enable Procedure Name Lookup field.

Field	Description/Action
JDBC Type	Use the Insert Row icon (or the Fill in all rows to the table icon) and specify the JDBC type of the corresponding return field for the stored procedure.
Return Field Name	Use the Insert Row icon (or the Fill in all rows to the table icon) to add return field names for the stored procedure.

8 Use the Parameter tab to specify the stored procedure's parameters.

9 Use the Insert Row icon (or the Fill in all rows to the table icon) to create new stored procedure parameters as needed.

Field	Description/Action
Param JDBC Type	The JDBC type of the stored procedure parameter.
Param Name	The stored procedure parameter name.
Param Type	Define the parameter type as IN, INOUT, or OUT.
Expression	The default value is ?, which acts as a placeholder for the variable so that you can set the input variable for that column at run time, or get input external to this adapter service. It adds one row with the same column name to the table. You can also type a fixed value as input now or at run time. If you choose to type a fixed value, you type a stored procedure call statement with values you set using this field.
Input Name	The name of any input parameters.
Input Type	The input parameter Java type. For a list of JDBC type to Java type mappings, see "JDBC Data Type to Java Data Type Mappings" on page 170.
Output Name	The name of any output parameters.
Output Type	The output parameter Java type. For a list of JDBC type to Java type mappings, see "JDBC Data Type to Java Data Type Mappings" on page 170.

10 If the procedure returns a result set, select the ResultSet tab to specify result set parameters using the fields in the following table.

StoredProcedure services can support multiple results sets. Use the Insert Row icon  to create additional result sets as needed. Use the following fields:

Field	Description/Action
Result Set Index	An index is automatically assigned to each result set. The first row default value is 1.
Result Set Name	The name of the result set you want to create.
Result Set Name (from second row)	Select result set name.
Column Name	The name of the column of the result set.
JDBC Type	The JDBC type of the result column.
Output Type	The Java type of the result column. For a list of JDBC type to Java type mappings, see "JDBC Data Type to Java Data Type Mappings" on page 170.

11 For information about using the Input/Output, Settings, Audit, Permissions, and Results tabs, see the *webMethods Developer User's Guide*. These tabs apply to all services that you configure using versions of the Developer earlier than 6.1.

When using Developer 6.1 or later, the Developer only contains the Adapter Settings and Input/Output tabs. The information from the Audit and Permissions tabs appears in the Properties panel, and the information from the Results tab appears in the Results panel.

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

To run or test the service directly, see "Testing Adapter Services" on page 119.

Testing Adapter Services

You use the Developer to test adapter services.

If you are using Integration Server 6.1 or later, make sure you are viewing the webMethods Developer in the Test perspective, as described in "Viewing Different Perspectives of the webMethods Developer" on page 43.

To test adapter services in the Developer Adapter Service Editor

1 In the Developer Service Browser, expand the package and folder that contain the service you want to test.

2 Select the service you want to test.

The Developer displays the configured adapter 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 If you use a version of the Developer earlier than 6.1, click the Results tab to view the output from this service. If you use Developer 6.1 or later, view the Results panel.

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 Developer 6.1 or later, make sure you are viewing the webMethods Developer in the Edit perspective, as described in "Viewing Different Perspectives of the webMethods Developer" on page 43.

To view a service

1 In the Developer Service Browser, expand the package and folder that contain the service you want to view.

2 Select the service you want to view.

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

Editing Adapter Services

You use the Developer to edit adapter services. If you are using Developer 6.1 or later, make sure you are viewing the webMethods Developer in the Edit perspective, as described in "Viewing Different Perspectives of the webMethods Developer" on page 43.

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 versions of the Integration Server earlier than 6.1, you *cannot* change which connection an adapter service uses after the service is configured.
- When using Integration Server 6.1 or later, 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 19.

To edit a service

1. In the Developer Service Browser, expand the package and folder that contain the service you want to view.
2. Select the service you want to edit.

   The Developer displays the configured adapter service in the service template's Adapter Service Editor.
3. Modify the values for 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.

Note: Because adapter services inherently depend on connections, you cannot change an adapter connection for an adapter service after you configure it.

Deleting Adapter Services

You use the Developer to delete adapter services. If you are using Developer 6.1 or later, make sure you are viewing the webMethods Developer in the Edit perspective, as described in "Viewing Different Perspectives of the webMethods Developer" on page 43.

To delete a service

1. In the Developer Service Browser, expand the package and folder that contain the service you want to delete.
2. Right-click the adapter service and then click Delete.

Validating Adapter Service Values

The Developer enables the JDBC Adapter to validate user-defined data for adapter services at design time. You can validate the values for a single adapter service or you can configure the Developer to always validate the values for adapter services.

Note: If you select the option to always validate 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 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.

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. With versions of the Integration Server earlier than 6.1, the icon is located on the Developer tool bar. With the Integration Server 6.1 or later, the icon is located on the editor panel.

The Automatic data validation 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. 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 JDBC 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 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 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.

This option is also available as an icon on the Developer toolbar. With versions of the Integration Server earlier than 6.1, this icon (the Refresh icon) is located on the

Developer tool bar. With the Integration Server 6.1 or later, the icon (Reload Values from the Adapter icon ) is located on the editor panel.

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.

webMethods

CHAPTER 5

Adapter Notifications

Overview

This chapter describes how to configure and manage JDBC Adapter notifications. For detailed descriptions of the available JDBC Adapter notifications, see "Adapter Notifications" on page 22.

Before Configuring or Managing Notifications

To prepare to configure or manage JDBC Adapter notifications

1. Start your Integration Server and the Administrator, if they are not already running.
2. Make sure you have Integration Server Administrator Developer privileges so that you can access the JDBC Adapter's administrative screens. See the *webMethods Integration Server Administrator's Guide* for information about setting user privileges.
3. Be sure to check "Database Driver Known Limitations" on page 187 for a list of known limitations for your database driver since it may affect how you configure your connections and notifications.
4. If you have made changes to the table schema for a given adapter notification, be sure to update the adapter notification accordingly.
5. If you plan to use the Only Once notification feature, see "Using the Exactly Once Notification Feature" on page 152 for details.
6. Using the Administrator, make sure the WmJDBCAdapter package is enabled. See "Enabling and Disabling Packages" on page 48 for instructions.
7. Using the Administrator, configure an adapter connection to use with the notification. See "Configuring JDBC Adapter Connections" on page 68 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.

Beginning with version 6.1, the Integration Server solves this limitation by providing a built-in service 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 19.

8. Start the Developer if it is not already running.
9. If you are using the Developer 6.1 or later, make sure you are viewing the webMethods Developer in the Edit perspective, as described in "Viewing Different Perspectives of the webMethods Developer" on page 43.

10 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 WmJDBCAdapter package. For more information about managing packages for the adapter, see Chapter 2, "Package Management" on page 45.

11 You must schedule a notification and then enable it before you can use the notification. See "Managing Polling Notifications" on page 149 for instructions.

Configuring InsertNotifications

An InsertNotification publishes notification of insert operations on a database table. You configure notifications using the Developer. For more information about notifications, see "Adapter Notifications" on page 22.

Be sure to review the section "Before Configuring or Managing Notifications" on page 124 before you configure notifications.

Note: InsertNotifications are not supported using Teradata databases.

To configure an InsertNotification

1 From the Developer File menu, select New...

2 Select Adapter Notification from the list of elements and click Next.

3 Select JDBC Adapter as the adapter type and click Next.

4 Select the appropriate Adapter Connection Name and click Next.

5 Select the InsertNotificaton template and click Next.

6 Type a unique name for the notification and select the appropriate folder. Click Next.

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 22. For more details about the 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 or later, see the *Publish-Subscribe Developer's Guide*.

8 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.

9 Select the Notification Configure tab and use the following fields:

Field	Description/Action
Base Name	The base name used to generate the Resource Name created by the JDBC Adapter. Note: For OS/390 DB2 7.2, the Base Name you create below must be no more than 5 characters because triggers on OS/390 name cannot be more than 8 characters.
Resource Type	Types are buffer table, trigger, and sequence. The base name and resource type determine the following Resource Name.
Resource Name	To ensure uniqueness, the resource name combines the following elements. You do not edit this name. ■ Resource prefix (WMB, WMT, and WMS for buffer table, trigger, and sequence respectively) ■ The name you typed in the Base Name field ■ A suffix, based on a system timestamp
File Record Format	The format of the file record. Optional field used by DB2 for AS/400 V4R5 only.
Database Name	The name of the database where the buffer tables will be created. Optional field used by DB2 for OS/390 only.
Table Space Name	The table space where the buffer tables will be created. Optional field used by DB2 for OS/390 only.

10 Select the Tables tab and use the following fields:

Note: For AS/400 DB2 V4R5 using a jt400.jar file, the table name for the notification cannot exceed 10 characters; otherwise, an exception will be generated when you try to enable the notification.

Field	Description/Action
Table Alias	The table alias is automatically assigned when you select more than one table in the Table Name field. The default is `t1`.
Table Name	Select a table name. The default for the associated catalog name is `current catalog`. The default for the associated schema name is `current schema`. The table name must not contain a period. If the table name does contain a period, Developer will throw a Java exception. Note: Informix databases do not allow you to specify a catalog and database name because you can only access the current catalog.
Type	The table type displays automatically based on the table you select.

11 If you are not joining tables, skip this step. Select the Joins tab to specify the columns for joining the tables you just configured.

a Select the Insert Row icon to create new left and right columns.

b Select Left Column and select the first table's joining column.

c Select the appropriate Operator.

d Select Right Column and select the next table's joining column.

e Repeat until you have defined all the joins.

12 Use the SELECT tab to define the columns and fields to be selected using the following fields:

a In the ALL/DISTINCT field, select ALL to include duplicate rows or DISTINCT to suppress duplicate rows. Selecting ALL corresponds to the SQL statement `SELECT ALL name from tablename`. The default value is blank, which corresponds to the SQL statement `SELECT name from tablename`.

b Select the Insert Row icon (or the Fill in all rows to the table icon) to create new fields as needed.

c In the Expression field, select a column or type any valid SQL expression. The corresponding Column Type, JDBC Type, Output Field Type, and Output Field display for each column you select in the Expression field. Use the following fields:

Note: You can use the BigDecimal data type with the JDBC Adapter. However, the webMethods Developer does not support the BigDecimal data type. This means that the Developer will not correctly display a BigDecimal data type result and you cannot enter a value of this data type. However, the JDBC Adapter will process the BigDecimal data type correctly.

Field	Description/Action
Expression	The column name or SQL expression.
Column Type	The column data type defined in the database table.
JDBC Type	The JDBC type of the corresponding Output Field.
Output Field Type	The data type of the output field. The JDBC Adapter automatically converts database-specific types to Java data types. For a list of JDBC type to Java type mappings, see "JDBC Data Type to Java Data Type Mappings" on page 170.
Output Field	The name of the field containing the output from the SELECT operation. An output field name displays when you select an expression. You can also modify the output field names as required.
Maximum Row	Specifies the number of rows to be retrieved from the buffer table. This field is useful when you are working with a large number of records and you want to limit the number of documents sent each time the notification polls. Use a value of `0` to indicate no limit on the number of rows retrieved.

13 Use the WHEN tab to specify the conditions for selecting information using the following table.

Note: If you use Microsoft SQL Server, Sybase, or V4 AS/400 DB2, do not use the WHEN tab because this feature is not supported. An exception will be generated if you try to use this tab.

a Select the Insert Row icon to define new WHEN clause fields.

b Select the Column field and choose a column from the list.

c Select a logical operator from the AND/OR field, an Operator, and separators (the left and right parentheses) as needed.

d Type a fixed value in the Value field. Be sure that it is a valid value, or an exception will be generated at run time.

e If necessary, use the Shift Up or Shift Down icons to change the order of the WHEN clause to ensure the parameters are parsed in the correct order.

f Repeat until you have specified all WHEN parameters.

14 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 or later, the information from the Permissions tab now appears in the Properties panel.

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

16 You must schedule and enable the notification using the Server Administrator before you can use it. See "Managing Polling Notifications" on page 149 for details.

Configuring UpdateNotifications

An UpdateNotification publishes notification of update operations on a database table. You configure notifications using the Developer. For more information about notifications, see "Adapter Notifications" on page 22.

Be sure to review the section "Before Configuring or Managing Notifications" on page 124 before you configure notifications.

Note: UpdateNotifications are not supported using Teradata databases.

To configure an UpdateNotification

1 From the Developer File menu, select New...

2 Select Adapter Notification from the list of elements and click Next.

3 Select JDBC Adapter as the adapter type and click Next.

4 Select the appropriate Adapter Connection Name and click Next.

5 Select the UpdateNotificaton template and click Next.

6 Type a unique name for the notification and select the appropriate folder. Click **Next**.

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 22. For more details about the 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 or later, see the *Publish-Subscribe Developer's Guide*.

8 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.

9 Select the **Notification Configure** tab and use the following fields:

Field	Description/Action
Base Name	The base name used to generate the **Resource Name** created by the JDBC Adapter. **Note:** For OS/390 DB2V7.2, the **Base Name** you create below must be no more than 5 characters because triggers on OS/390 name cannot be more than 8 characters.
Resource Type	Types are buffer table, trigger, and sequence. The base name and resource type determine the following **Resource Name**.
Resource Name	To ensure uniqueness, the resource name combines the following elements. You do not edit this name. ■ Resource prefix (WMB, WMT, and WMS for buffer table, trigger, and sequence respectively) ■ The name you typed in the **Base Name** field ■ A suffix, based on a system timestamp
File Record Format	The format of the file record. Optional field used by DB2 for AS/400 V4R5 only.
Database Name	The name of the database where the buffer tables will be created. Optional field used by DB2 for OS/390 only.
Table Space Name	The table space where the buffer tables will be created. Optional field used by DB2 for OS/390 only.

10 Select the Tables tab and use the following fields:

Note: For AS/400 DB2 V4R5 using a jt400.jar file, the table name for the notification cannot exceed 10 characters; otherwise, an exception will be generated when you try to enable the notification.

Field	Description/Action
Table Alias	The table alias is automatically assigned when you select more than one table in the Table Name field. The default is `t1`.
Table Name	Select a table. The default for the associated catalog name is `current catalog`. The default for the associated schema name is `current schema`. The table name must not contain a period. If the table name does contain a period, Developer will throw a Java exception. Note: Informix databases do not allow you to specify a catalog and database name because you can only access the current catalog.
Type	The table type displays automatically based on the table you select.

11 If you are not joining tables, skip this step. Select the Joins tab to specify the columns for joining the tables you just configured.

a Select the Insert Row icon to create new left and right columns.

b Select Left Column and select the first table's joining column.

c Select the appropriate Operator.

d Select Right Column and select the next table's joining column.

e Repeat until you have defined all the joins.

12 Use the SELECT tab to define the columns and fields to be selected as follows:

a In the ALL/DISTINCT field, select ALL to include duplicate rows or DISTINCT to suppress duplicate rows. Selecting ALL corresponds to the SQL statement `SELECT ALL name from tablename`. The default value is blank, which corresponds to the SQL statement `SELECT name from tablename`.

b Select the Insert Row icon (or the Fill in all rows to the table icon) to create new fields as needed.

c In the Expression field, select a column or type any valid SQL expression. The corresponding Column Type, JDBC Type, Output Field Type, and Output Field display for each column you select in the Expression field. Use the following fields:

Note: You can use the BigDecimal data type with the JDBC Adapter. However, the webMethods Developer does not support the BigDecimal data type. This means that the Developer will not correctly display a BigDecimal data type result and you cannot enter a value of this data type. However, the JDBC Adapter will process the BigDecimal data type correctly.

Field	Description/Action
Expression	The column name or SQL expression.
Column Type	The column data type defined in the database table.
JDBC Type	The JDBC type of the corresponding Output Field.
Output Field Type	The data type of the output field. The JDBC Adapter automatically converts database-specific types to Java data types. For a list of JDBC type to Java type mappings, see "JDBC Data Type to Java Data Type Mappings" on page 170.
Output Field	The name of the field containing the output from the SELECT operation. An output field name displays when you select an expression. You can also modify the output field names as required.
Notify On Update	Enable this option to indicate which of the columns specified in the SELECT tab you want notification if updated. Select: ■ Yes: you want notification if this column of data has been updated ■ No: you do not want notification if this column of data has been updated For example, you configure the following three output fields: MyName, MyNumber, and MyLocation. You want notification only if the MyLocation output field is updated. In this case, you would select Yes for the MyLocation output field, and select No for the MyName and MyNumber output fields.

Field	Description/Action
Maximum Row	Specifies the number of rows to be retrieved from the buffer table. This field is useful when you are working with a large number of records and you want to limit the number of documents sent each time the notification polls. Use a value of `0` to indicate no limit on the number of rows retrieved.

13 Use the WHEN tab to specify the conditions for selecting information:

Note: If you use Microsoft SQL Server, Sybase, or V4 AS/400 DB2, do not use the WHEN tab because this feature is not supported. An exception will be generated if you try to use this tab.

a Select the Insert Row icon to define new WHEN clause fields.

b Select the Column field and choose a column from the list.

c Select a logical operator from the AND/OR field, an Operator, and separators (the left and right parentheses) as needed.

d Type a fixed value in the Value field. Be sure that it is a valid value, or an exception will be generated at run time.

e If necessary, use the Shift Up or Shift Down icons to change the order of the WHEN clause to ensure the parameters are parsed in the correct order.

f Repeat until you have specified all WHEN parameters.

14 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 or later, the information from the Permissions tab appears in the Properties panel.

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

16 You must schedule and enable the notification using the Server Administrator before you can use it. See "Managing Polling Notifications" on page 149 for details.

Configuring DeleteNotifications

A DeleteNotification publishes notification of delete operations on a database table. You configure notifications using the Developer. For more information about notifications, see "Adapter Notifications" on page 22.

Be sure to review the section "Before Configuring or Managing Notifications" on page 124 before you configure notifications.

Note: DeleteNotifications are not supported using Teradata databases.

To configure a DeleteNotification

1 From the Developer File menu, select New...

2 Select Adapter Notification from the list of elements and click Next.

3 Select JDBC Adapter as the adapter type and click Next.

4 Select the appropriate Adapter Connection Name and click Next.

5 Select the DeleteNotificaton template and click Next.

6 Type a unique name for the notification and select the appropriate folder. Click Next.

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 22. For more details about the 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 or later, see the *Publish-Subscribe Developer's Guide*.

8 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.

9 Select the Notification Configure tab and use the following fields:

Field	Description/Action
Base Name	The base name used to generate the Resource Name created by the JDBC Adapter. Note: For OS/390 DB2V7.2, the Base Name you create below must be no more than 5 characters because triggers on OS/390 name cannot be more than 8 characters.
Resource Type	Types are buffer table, trigger, and sequence. The base name and resource type determine the following Resource Name.
Resource Name	To ensure uniqueness, the resource name combines the following elements. You do not edit this name. ■ Resource type prefix (WMB, WMT, and WMS for buffer table, trigger, and sequence respectively) ■ The name you typed in the Base Name field ■ A suffix, based on a system timestamp
File Record Format	The format of the file record. Optional field used by DB2 for AS/400 V4R5 only.
Database Name	The name of the database where the buffer tables will be created. Optional field used by DB2 for OS/390 only.
Table Space Name	The table space where the buffer tables will be created. Optional field used by DB2 for OS/390 only.

10 Select the Tables tab and use the following fields:

Note: For AS/400 DB2 V4R5 using a jt400.jar file, the table name for the notification cannot exceed 10 characters; otherwise, an exception will be generated when you try to enable the notification.

Field	Description/Action
Table Alias	The table alias is automatically assigned when you select more than one table in the Table Name field. The default is `t1`.
Table Name	Select a table name. The default for the associated catalog name is `current catalog`. The default for the associated schema name is `current schema`. The table name must not contain a period. If the table name does contain a period, Developer will throw a Java exception. Note: Informix databases do not allow you to specify a catalog and database name because you can only access the current catalog.
Type	The table type displays automatically based on the table you select.

11 If you are not joining tables, skip this step. Select the Joins tab to specify the columns for joining the tables you just configured.

a Select the Insert Row icon to create new left and right columns.

b Select Left Column and select the first table's joining column.

c Select the appropriate Operator.

d Select Right Column and select the next table's joining column.

e Repeat until you have defined all the joins.

12 Use the SELECT tab to define the columns and fields to be selected.

a In the ALL/DISTINCT field, select ALL to include duplicate rows or DISTINCT to suppress duplicate rows. Selecting ALL corresponds to the SQL statement `SELECT ALL` *`name`* `from` *`tablename`*. The default value is blank, which corresponds to the SQL statement `SELECT` *`name`* `from` *tablename*.

b Select the Insert Row icon (or the Fill in all rows to the table icon) to create new fields as needed.

c In the Expression field, select a column or type any valid SQL expression. The corresponding Column Type, JDBC Type, Output Field Type, and Output Field display for each column you select in the Expression field. Use the following fields:

Note: You can use the BigDecimal data type with the JDBC Adapter. However, the webMethods Developer does not support the BigDecimal data type. This means that the Developer will not correctly display a BigDecimal data type result and you cannot enter a value of this data type. However, the JDBC Adapter will process the BigDecimal data type correctly.

Field	Description/Action
Expression	The column name or SQL expression.
Column Type	The column data type defined in the database table.
JDBC Type	The JDBC type of the corresponding Output Field.
Output Field Type	The data type of the output field. The JDBC Adapter automatically converts database-specific types to Java data types. For a list of JDBC type to Java type mappings, see "JDBC Data Type to Java Data Type Mappings" on page 170.
Output Field	The name of the field containing the output from the select operation. An output field name displays when you select an expression. You can also modify the output field names as required.
Maximum Row	Specifies the number of rows to be retrieved from the buffer table. This field is useful when you are working with a large number of records and you want to limit the number of documents sent each time the notification polls. Use a value of `0` to indicate no limit on the number of rows retrieved.

13 Use the WHEN tab to specify the conditions for selecting information:

Note: If you use Microsoft SQL Server, Sybase, or V4 AS/400 DB2, do not use the WHEN tab because this feature is not supported. An exception will be generated if you try to use this tab.

a Select the Insert Row icon to define new WHEN clause fields.

b Select the Column field and choose a column from the list.

c Select a logical operator from the AND/OR field, an Operator, and separators (the left and right parentheses) as needed.

d Type a fixed value in the Value field. Be sure that it is a valid value, or an exception will be generated at run time.

e If necessary, use the Shift Up or Shift Down icons to change the order of the WHEN clause to ensure the parameters are parsed in the correct order.

f Repeat until you have specified all WHEN parameters.

14 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 or later, the information from the Permissions tab appears in the Properties panel.

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

16 You must schedule and enable the notification using the Server Administrator before you can use it. See "Managing Polling Notifications" on page 149 for details.

Configuring BasicNotifications

A BasicNotification polls a database table for data using a SQL Select operation. You configure notifications using the Developer. For more information about notifications, see "Adapter Notifications" on page 22.

Be sure to review the section "Before Configuring or Managing Notifications" on page 124 before you configure notifications.

Note: Running a BasicNotification may generate a *duplicate message* error; the Integration Server will ignore the duplicate notification document. In this case, you should check the Delete selected records option on the SELECT tab (described in step 11 below) and choose a column with sequentially unique values as the Record ID Column value.

To configure a BasicNotification

1 From the Developer File menu, select New...

2 Select Adapter Notification from the list of elements and click Next.

3 Select JDBC Adapter as the adapter type and click Next.

4 Select the appropriate Adapter Connection Name and click Next.

5 Select the BasicNotificaton template and click Next.

6 Type a unique name for the notification and select the appropriate folder. Click Next.

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 22. For more details about the 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 or later, see the *Publish-Subscribe Developer's Guide*.

8 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.

9 Select the Tables tab and use the following fields:

Note: For AS/400 DB2 V4R5 using a jt400.jar file, the table name for the notification cannot exceed 10 characters; otherwise, an exception will be generated when you try to enable the notification.

Field	Description/Action
Table Alias	The table alias is automatically assigned when you select more than one table in the Table Name field. The default is `t1`.
Table Name	Select a table name. The default for the associated catalog name is `current catalog`. The default for the associated schema name is `current schema`. The table name must not contain a period. If the table name does contain a period, Developer will throw a Java exception. Note: Informix databases do not allow you to specify a catalog and database name because you can only access the current catalog.
Type	The table type displays automatically based on the table you select.

10 If you are not joining tables, skip this step. Select the Joins tab to specify the columns for joining the tables you just configured.

a Select the Insert Row icon (or the Fill in all rows to the table icon) to create new left and right columns.

b Select Left Column and select the first table's joining column.

c Select the appropriate Operator.

d Select Right Column and select the next table's joining column.

e Repeat until you have defined all the joins.

11 Use the SELECT tab to define the columns and fields to be selected.

a In the ALL/DISTINCT field, select ALL to include duplicate rows or DISTINCT to suppress duplicate rows. Selecting ALL corresponds to the SQL statement `SELECT ALL` *`name`* `from` *`tablename`*. The default value is blank, which corresponds to the SQL statement `SELECT` *`name`* `from` *tablename*.

b Select the Insert Row icon  (or the Fill in all rows to the table icon ) to create new fields as needed.

c In the Expression field, select a column or type any valid SQL expression. The corresponding Column Type, JDBC Type, Output Field Type, and Output Field display for each column you select in the Expression field. Use the following fields:

Note: You can use the BigDecimal data type with the JDBC Adapter. However, the webMethods Developer does not support the BigDecimal data type. This means that the Developer will not correctly display a BigDecimal data type result and you cannot enter a value of this data type. However, the JDBC Adapter will process the BigDecimal data type correctly.

Field	Description/Action
Expression	The column name or SQL expression.
Column Type	The column data type defined in the database table.
JDBC Type	The JDBC type of the corresponding Output Field.
Output Field Type	The data type of the output field. The JDBC Adapter automatically converts database-specific types to Java data types. For a list of JDBC type to Java type mappings, see "JDBC Data Type to Java Data Type Mappings" on page 170.
Output Field	The name of the field containing the output from the SELECT operation. An output field name displays when you select an expression. You can also modify the output field names as required.

Field	Description/Action
Sort Order	Specifies ordering of publishable documents per each polling. Use this field to ensure that the notification's publishable documents, for each polling, are in the correct ascending or descending order based on one or more table columns. Select either Ascend or Descend. Leave the field blank if there is no sort order.

d If you want to use the Exactly Once notification feature, you must enable the Exactly Once Notification option. See "Configuring InsertNotifications" on page 125 for more information.

e Set the Delete selected records flag to automatically delete the selected records from the buffer table (based on their Record ID Column value as entered in step f) after the notification. Use this option to prevent publishing the same documents to the Integration Server each time polling occurs.

You must enable the Delete selected records option to use the Exactly Once notification feature. See "Configuring InsertNotifications" on page 125 for more information.

Note: Running a BasicNotification may generate a *duplicate message* error; the Integration Server will ignore the duplicate notification document. In this case, you should check the Delete selected records option and choose a column with sequentially unique values as the Record ID Column value in step f below.

f You must use the Record ID Column field to use the Exactly Once notification feature. Select the column from the buffer table that you want to use as the unique ID for the publishable document for this notification. See "Configuring InsertNotifications" on page 125 for more details.

To ensure that all values will be unique, choose a table column in the Record ID Column field whose values are sequential numbers.

g Use the Maximum Row field to specify the maximum number of records to retrieve from the database. This field is useful when you are working with a large number of records and you want to limit the number of documents sent each time the notification polls.

The default value of `0` (no limit) retrieves all records.

12 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 or later, the information from the Permissions tab appears in the Properties panel.

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

14 You must schedule and enable the notification using the Server Administrator before you can use it. See "Managing Polling Notifications" on page 149 for details.

Configuring StoredProcedureNotifications

A StoredProcedureNotification publishes notification data by calling a stored procedure inside of a database. You configure notifications using the Developer. For more information about notifications, see "Adapter Notifications" on page 22.

Be sure to review the section "Before Configuring or Managing Notifications" on page 124 before you configure notifications.

For details and important considerations when using a StoredProcedureNotification, see "Stored Procedure Notifications" on page 31.

To configure a StoredProcedureNotification

1 From the Developer File menu, select New...

2 Select Adapter Notification from the list of elements and click Next.

3 Select JDBC Adapter as the adapter type and click Next.

4 Select the appropriate Adapter Connection Name and click Next.

5 Select the StoredProcedureNotificaton template and click Next.

6 Type a unique name for the notification and select the appropriate folder. Click Next.

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 22. For more details about the 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 or later, see the *Publish-Subscribe Developer's Guide*.

8 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.

9 Select the Call tab to specify which stored procedure to use with the notification. Use the Insert Row icon and set the Call parameters as follows:

Field	Description/Action
Catalog Name	The name of the catalog. The default for the catalog name is `current catalog`.
Schema Name	The name of the schema. The default for the schema name is `current schema`.
Enable Procedure Lookup (Optional)	To type in the Procedure Name, set this field to False. To select the Procedure Name from a list, set this field to True. The default is False. Set this value to False if you know the name of the procedure and you are working with a large database that has a long list of procedures.
Procedure Name	Type or select the stored procedure name, depending on how you set the Enable Procedure Lookup field.
JDBC Type	The JDBC type of the corresponding Return Field Name.
Return Field Name	Name of the return field of the stored procedure.

10 Use the Parameter tab to specify stored procedure parameters. Use the Insert Row icon (or the Fill in all rows to the table icon) to create new parameters for the stored procedure.

Note: You can use the BigDecimal data type with the JDBC Adapter. However, the webMethods Developer does not support the BigDecimal data type. This means that the Developer will not correctly display a BigDecimal data type result and you cannot enter a value of this data type. However, the JDBC Adapter will process the BigDecimal data type correctly.

Field	Description/Action
ParamJDBCType	The JDBC type of the stored procedure parameter. For a list of JDBC type to Java type mappings, see "JDBC Data Type to Java Data Type Mappings" on page 170.
ParamName	Stored procedure parameter name.
ParamType	Select OUT as the parameter type because StoredProcedure Notifications do not accept input parameters.

Field	Description/Action
Expression	Keep the default value of ? because StoredProcedure Notifications do not accept input parameters.
Output Name	Name of any output parameters of the stored procedure, if any. See "Stored Procedure Notifications" on page 31 for information about output fields for stored procedures.
Output Type	Output parameter Java type. For a list of JDBC type to Java type mappings, see "JDBC Data Type to Java Data Type Mappings" on page 170.

11 StoredProcedure notifications can support one result set (or one Oracle REF CURSOR). If the procedure returns a result set, select the ResultSet tab to specify result set parameters using the fields in the following table:

Field	Description/Action
Result Set Index	An index is automatically assigned to each result set. The first row default value is 1.
Result Set Name	Type the name of the result set you want to create. See "Stored Procedure Notifications" on page 31 for information about result sets.
Result Set Name (from second row)	Select a valid result set name.
Column Name	Name of column of the result set.
JDBC Type	The JDBC type of the result set column.
Output Type	The Java type of the result column. For a list of JDBC type to Java type mappings, see "JDBC Data Type to Java Data Type Mappings" on page 170.

12 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 or later, the information from the Permissions tab appears in the Properties panel.

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

14 You must schedule and enable the notification using the Server Administrator before you can use it. See "Managing Polling Notifications" on page 149 for details.

Configuring OrderedNotifications

An OrderedNotification publishes notification data for multiple insert, update, or delete operations on multiple tables. You configure notifications using the Developer. For more information about notifications, see "Adapter Notifications" on page 22.

Be sure to review the section "Before Configuring or Managing Notifications" on page 124 before you configure notifications.

Note: OrderedNotifications are not supported using TeradataV2R5 databases.

With OrderedNotifications, typically you configure an Integration Server trigger to subscribe to the notification's publishable document and a flow service that the trigger invokes. Because the primary reason to use OrderedNotifications is to preserve the order in which the operations occur, be sure to use the **Process Document Serially** option on the **Settings** tab in Developer when you create the trigger and flow service. For more information about using configuring Integration Servers triggers and flow services, see the *webMethods Developer User's Guide*.

To configure an OrderedNotification

1. From the Developer **File** menu, select **New...**
2. Select **Adapter Notification** from the list of elements and click **Next**.
3. Select **JDBC Adapter** as the adapter type and click **Next**.
4. Select the appropriate **Adapter Connection Name** and click **Next**.
5. Select the **StoredProcedureNotificaton** template and click **Next**.
6. Type a unique name for the notification and select the appropriate folder. Click **Next**.
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 22. For more details about the 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 or later, see the *Publish-Subscribe Developer's Guide*.
8. The editor for the adapter notification appears. You can select the **Adapter Settings** tab at any time to confirm notification properties such as the **Adapter Name**, **Adapter Connection Name**, and **Adapter Notification Template**, as necessary.

9 Select the Notification Configure tab and use the following fields:

Field	Description/Action
Base Name	The base name used to generate the Resource Name created by the JDBC Adapter. Note: For OS/390 DB2V7.2, the Base Name you create below must be no more than 5 characters because triggers on OS/390 name cannot be more than 8 characters.
Resource Type	Types are buffer table, trigger, and sequence. The base name and resource type determine the following Resource Name.
Resource Name	To ensure uniqueness, the resource name combines the following elements. You do not edit this name. ■ Resource prefix (WMB, WMT, and WMS for buffer table, trigger, and sequence respectively) ■ The name you typed in the Base Name field ■ A suffix, based on a system timestamp
File Record Format	The format of the file record. Optional field used by DB2 for AS/400 V4R5 only.
Database Name	The name of the database where the buffer tables will be created. Optional field used by DB2 for OS/390 only.
Table Space Name	The table space where the buffer tables will be created. Optional field used by DB2 for OS/390 only.

10 Select the Source Tables tab and use the following fields:

Note: For AS/400 DB2 V4R5 using a jt400.jar file, the table name for the notification cannot exceed 10 characters; otherwise, an exception will be generated when you try to enable the notification.

Field	Description/Action
Table Alias	The table alias is automatically assigned when you select more than one table in the Table Name field. The default is `t1`.
Table Name	Select a table name. The default for the associated catalog name is `current catalog`. The default for the associated schema name is `current schema`. The table name must not contain a period. If the table name does contain a period, Developer will throw a Java exception. Note: Informix databases do not allow you to specify a catalog and database name because you can only access the current catalog.
Type	The table type displays automatically based on the table you select.
Operation Type	Select INSERT, UPDATE, or DELETE operation.
Operation ID	Assign an ID to uniquely identify the given operation for the notification.

11 Use the SELECT tab to define the columns and fields to be selected using the following fields:

a In the ALL/DISTINCT field, select ALL to include duplicate rows or DISTINCT to suppress duplicate rows. Selecting ALL corresponds to the SQL statement `SELECT ALL` *`name`* `from` *`tablename`*. The default value is blank, which corresponds to the SQL statement `SELECT` *`name`* `from` *`tablename`*.

b Select the Insert Row icon (or the Fill in all rows to the table icon) to create new fields as needed. For each Expression column you select, the corresponding Operation ID, Column Type, JDBC Type, Output Field Type, and Output Field display.

Note: You can use the BigDecimal data type with the JDBC Adapter. However, the webMethods Developer does not support the BigDecimal data type. This means that the Developer will not correctly display a BigDecimal data type result and you cannot enter a value of this data type. However, the JDBC Adapter will process the BigDecimal data type correctly.

Use the following fields:

Field	Description/Action
Expression	The column name.
Operation ID	The corresponding operation ID for the expression.
Column Type	The column data type defined in the database table.
JDBC Type	The JDBC type of the corresponding Output Field.
Output Field Type	The data type of the output field. The JDBC Adapter automatically converts database-specific types to Java data types. For a list of JDBC type to Java type mappings, see "JDBC Data Type to Java Data Type Mappings" on page 170.
Output Field	The name of the field containing the output from the SELECT operation. An output field name displays when you select an expression. You can also modify the output field names as required.
Notify On Update	Used for Update operations only. Enable this option to indicate which of the columns specified in the SELECT tab you want notification if updated. Select: ■ Yes: you want notification if this column of data has been updated ■ No: you do not want notification if this column of data has been updated For example, you configure the following three output fields: MyName, MyNumber, and MyLocation. You want notification only if the MyLocation output field is updated. In this case, you would select Yes for the MyLocation output field, and select No for the MyName and MyNumber output fields.

12 Use the WHEN tab to specify the conditions for selecting information using the following table.

Note: If you use Microsoft SQL Server or Sybase, do not use the WHEN tab because this feature is not supported. An exception will be generated if you try to use this tab.

a Select the Insert Row icon to define new WHEN clause fields.

b Select the Column field and choose a column from the list. The Operation ID will display after you make your selection.

c Select a logical operator from the AND/OR field, an Operator, and separators (the left and right parentheses) as needed.

d Type a fixed value in the Value field. Be sure that it is a valid value, or an exception will be generated at run time.

e If necessary, use the Shift Up or Shift Down icons to change the order of the WHEN clause to ensure the parameters are parsed in the correct order.

f Repeat until you have specified all WHEN parameters.

13 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 or later, the information from the Permissions tab appears in the Properties panel.

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

15 You must schedule and enable the notification using the Server Administrator before you can use it. See "Managing Polling Notifications" on page 149 for details.

Managing Polling Notifications

You must schedule a notification and then enable it before you can use the notification. Use the Integration Server Administrator along with the following procedures to do so.

Note: You must have webMethods administrator privileges to access the JDBC 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 JDBC Adapter.

3 From the navigation area, select Polling Notifications.

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

Note: For AS/400 DB2 V4R5 using a jt400.jar file, the table name for the notification cannot exceed 10 characters; otherwise, an exception will be generated when you try to enable the notification.

Note: If you use an XA-Transaction connection, you cannot enable a notification.

Field	Description/Action
Notification Name	The name of the notification.
Package Name	The name of the package for the notification.
Enabled	*This field applies only to Integration Server versions earlier than 6.5.* Note: You must schedule a polling notification before you can enable it. To schedule a polling notification, use the Edit Schedule  icon described in these procedures. 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. Enabling and disabling a notification affects how its trigger and buffer tables are created and dropped. For details, see "Polling Notifications and States" on page 41. If there is no polling notification scheduled for a given adapter notification, Not Scheduled appears in this field. Use the Edit Schedule  icon to create a polling notification as described in step 5.

Field	Description/Action
State	*This field applies only to Integration Server version 6.5.* Note: You must schedule a polling notification before you can enable it. To schedule a polling notification, use the Edit Schedule  icon described in these procedures. After you schedule a polling notification, you can use this option's dropdown list to set the polling notification's state: ■ Enabled: The polling notification performs as scheduled. ■ Suspended: The polling notification is removed from the scheduler but the database trigger and buffer table are not dropped. ■ Disabled: The polling notification is removed from the scheduler and the database trigger and buffer table are dropped. The Suspend all enabled and Resume all suspended links help you change states quickly for multiple polling notifications. Enabling, suspending, and disabling a notification affects how its trigger and buffer tables are created and dropped. For details, see "Polling Notifications and States" on page 41. If there is no polling notification scheduled for a given adapter notification, control for this field is disabled. Use the Edit Schedule  icon to create a polling notification as described in step 5.
Edit Schedule	Click on the Edit Schedule  icon to create or modify polling notification parameters. Note: You must disable a polling notification before you can edit it. Continue to step 5.
View Schedule	Click on the View Schedule icon to review the parameters for the selected polling notification. Click Return to JDBC Adapter Notifications to go back to the main polling notification page.

5 To create or modify schedule parameters for the selected adapter notification, click on the Edit Schedule  icon and use the following fields:

Field	Description/Action
Interval (seconds)	Type the polling interval time in seconds.
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.

6 Click Save Schedule.

7 After you create a polling notification, you can enable it. Use the State field described in step 4 to enable a polling notification.

Using the Exactly Once Notification Feature

Adapter notifications can use the Exactly Once notification feature. This feature ensures that notification data will not be duplicated even if a failure occurs during processing. This is achieved by assigning unique IDs for each publishable document. After a processing failure, the Integration Server checks for duplicate records in storage and ignores any duplicate IDs.

Because this feature ensures that the rows of the data in the buffer table will not be duplicated even after a processing failure, you should not re-create a notification in the event of a processing failure. The Exactly Once feature will automatically make the appropriate corrections as needed.

Note: Stored Procedure Notifications do not support the Exactly Once notification feature because they do not use publishable document unique IDs.

Enabling Exactly Once Notification

To use the Exactly Once feature, you must enable client-side queuing in the Integration Server. For details, see the *webMethods Integration Server Administrator's Guide*.

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.

With Insert Notifications, Update Notifications, and Delete Notifications, the buffer table and trigger remain in the database. When the Integration Server with the exported notifications starts, each configured notification starts to poll the data from the buffer table.

If you want to export configured notifications in a Disabled state, you need to disable the notifications before you export the package containing them. With Insert Notifications, Update Notifications, and Delete Notifications, the buffer table and trigger will be dropped when you disable the notification. When you enable the exported notification, the buffer table and trigger will be created.

You may export configured notifications in a Suspended state. The trigger and buffer table will not be dropped.

See "Insert Notifications, Update Notifications, and Delete Notifications" on page 24 for more details.

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 or later, make sure you are viewing the webMethods Developer in the Edit perspective, as described in "Viewing Different Perspectives of the webMethods Developer" on page 43.

To view a notification

1. In the Developer Service Browser, expand the package and folder that contain the notification you want to view.
2. 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 or later, make sure you are viewing the webMethods Developer in the Edit perspective, as described in "Viewing Different Perspectives of the webMethods Developer" on page 43.

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 or later, you *can* change which connection an adapter notification uses. To do this, you use the built-in service pub.art.notification:setPollingNotificationNodeConnection. For more information, see "Changing the Connection Associated with an Adapter Service or Notification at Design Time" on page 19.

To edit a notification

1. In the Developer Service Browser, expand the package and folder that contain the notification you want to view.
2. Select the notification you want to edit.

   The Developer displays the notification in the notification template's Adapter Notification Editor.
3. 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.

Note: Because adapter notification inherently depend on connections, you cannot edit or change the adapter connection for a notification after you configure it.

Deleting Notifications

You use the Developer to delete adapter notifications. If you are using the Developer 6.1 or later, make sure you are viewing the Developer in the Edit perspective, as described in "Viewing Different Perspectives of the webMethods Developer" on page 43.

Note: Before you delete the notification, be sure that you first disable it. Otherwise, the trigger and buffer table created by the notification will remain in the database. To disable a notification, see "Managing Polling Notifications" on page 149.

To delete a notification

1 In the Developer Service Browser, expand the package and folder that contain the notification you want to delete.

Right-click the notification and then click Delete.

Validating Adapter Notification Values

The Developer enables the JDBC 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 JDBC 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.

webMethods

CHAPTER 6

Logging and Exception Handling

Overview

The following sections describe message logging, JDBC Adapter exception handling, and customizing the JDBC Adapter's list of fatal error codes. A list of error codes and supporting information appears at the end of this chapter.

For a list of known database driver limitations, see "Database Driver Known Limitations" on page 187.

JDBC Adapter Message Logging

The JDBC Adapter uses the Integration Server logging mechanism to log messages. You can configure and view the Integration Server logs to monitor and troubleshoot the JDBC 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 JDBC 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 JDBC Adapter service template on the Audit tab.
Error Log	The JDBC 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 JDBC 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 log messages. V1-Verbose1 or V4-Verbose4 log messages appear as JDBC Adapter log messages.

The JDBC Adapter's log messages appear in the following format: ADA.1.nnnnc, where the facility code ADA indicates that the message is from an adapter, 1 indicates that it is the JDBC Adapter, nnnn represents the error's minor code, and (optionally) c represents the message's severity level. For detailed descriptions of the JDBC Adapter's minor codes, see "JDBC Adapter Error Codes" on page 161.

Because the JDBC 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 JDBC 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)

JDBC Adapter Exception Handling

The JDBC Adapter throws two kinds of exceptions that you should be aware of as you build integrations using the adapter: AdapterException and AdapterConnectionException. When creating a flow or Java service that incorporates an adapter service or notification, you might want to build logic into the wrapping service to catch and handle these types of exceptions.

AdapterException

The JDBC Adapter throws an AdapterException for two reasons:

1. To report an error related to the adapter's logic, such as a configuration error or a connection creation error.
2. To wrap an SQLException if the adapter does not consider the SQLException's SQLCODE to be a fatal error. In this case, WmART wraps the AdapterException in a com.wm.pkg.art.error.DetailedServiceException and throws it to the Integration Server. AdapterExceptions containing an error code of 316 are SQLExceptions.

To manage the AdapterException, you can catch the DetailedServiceException in a flow or Java service and then navigate through the nested exceptions to the AdapterException, which will contain the error code identifying the error.

AdapterConnectionException

The JDBC Adapter throws an AdapterConnectionException to wrap an SQLException if the adapter interprets the SQLCODE as a fatal error.

In this case:

- WmART 6.5 and later resets the entire connection pool.
- WmART earlier than 6.5 drops the connection from the connection pool.

WmART then wraps the exception in com.wm.pkg.art.error.DetailedSystemException and throws it to the Integration Server.

SQLException

When an adapter connection's associated JDBC driver fails to execute a SQL command against a database, the driver throws a SQLException. SQLExceptions include a SQL STATE, a SQLCODE, and an error message.

The JDBC Adapter catches the SQLException from the JDBC driver and, depending on the SQLCODE, wraps the SQLException in either an AdapterException or an AdapterConnectionException. If a SQL CODE is in the adapter's list of fatal errors for the database, the adapter wraps the exception in an AdapterConnectionException; otherwise, it wraps it in an AdapterException. Each AdapterException and AdapterConnectionException contains an adapter error code. If the error code is 316, then the exception wraps an SQLException.

Customizing the Adapter's List of Fatal Error Codes

You can add a specific error code to the list of fatal error codes. This allows the JDBC Adapter to automatically refresh JDBC Adapter connections when a specific error occurs. Be sure that there is no other use for this error code before you add it to the list.

To customize the fatal error list

1 Start the Integration Server Administrator if it is not already running.

2 Under **Settings** in the left panel, select **Extended**.

3 Select **Edit Extended Settings**. In the edit box, type:

```
watt.adapter.JDBC.database driver.fatalErrors=+ErrorCode_1,
ErrorCode_2, ErrorCode_n
```

Example: To allow the JDBC Adapter to refresh connections when encountering Oracle error codes 17002 and 17003 using an Oracle JDBC driver, type:

```
watt.adapter.JDBC.Oracle.fatalErrors=+17002, 17003
```

The following is a list of other supported driver settings (for watt.adapter.JDBC.database driver.fatalErrors):

Driver	Setting
Microsoft SQL Server	watt.adapter.JDBC.MsMssql.fatalErrors
Oracle JDBC	watt.adapter.JDBC.Oracle.fatalErrors
IBM DB2 Net	watt.adapter.JDBC.DB2NET.fatalErrors
IBM DB2 App	watt.adapter.JDBC.DB2APP.fatalErrors
JTOpen	watt.adapter.JDBC.DB2JTOPEN.fatalErrors
DataDirect Connect for JDBC driver for DB2	watt.adapter.JDBC.CJDBCDB2.fatalErrors
Teradata Type 4	watt.adapter.JDBC.TeraData.fatalErrors
JDBC 2.21 type 4 for Informix	watt.adapter.JDBC.INFORMIX.fatalErrors
jCONNECT 5.5 type 4 for Sybase	watt.adapter.JDBC.SYBASE.fatalErrors
DB2 Universal type 4	`watt.adapter.JDBC.DB2UNIVERSAL.fatalErrors`
Other driver types	watt.adapter.JDBC.Generic.fatalErrors

4 Click Save Changes.

5 Restart the JDBC Adapter.

JDBC Adapter Error Codes

The following table lists the JDBC Adapter's minor codes and provides information on the error message, reason, and possible action for each error.

---

200 The JDBC DataSource class ClassName cannot be located.

A DataSource class name was specified in the adapter Connection Properties DataSource Class field, but the class cannot be located.

Cause: 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/package/WmJDBCAdapter/code/jars directory.

201 The JDBC DataSource class ClassName cannot be instantiated.

Cause: The instantiation of the JDBC driver's DataSource class failed.

Response: Use a supported JDBC driver.

202 Cannot set properties for JDBC DataSource class ClassName.

Cause: Properties cannot be set through the DataSource class because the driver does not support the specified property.

Response: See "Configuring JDBC Adapter Connections", step 5 on page 69 for supported drivers and their settings.

203 The JDBC DataSource class ClassName does not have some of the configured property settings.

Cause: Some properties specified in the connection's properties are not correct.

Response: See "Configuring JDBC Adapter Connections", step 5 on page 69 for supported drivers and their settings.

204 Cannot connect to the database with DataSource class ClassName.

Cause: The connection between the adapter and the database failed.

Response: Check the SQL exception in the Integration Server error log, and check the database error messages.

205 Cannot retrieve the database metadata MetadataElement.

Cause: An error occurred when the adapter tried to retrieve database metadata information.

Response: Check the SQL exception in the Integration Server error log, and check the database error messages.

206 The JDBC DataSource class *ClassName* is not XADataSource.

Cause: The DataSource class name you specified in the Connection Properties DataSource Name field is not an XADataSource.

Response: See your JDBC Adapter documentation for supported drivers and DataSource class names.

207 The JDBC DataSource class *ClassName* does not support LOCAL_TRANSACTION.

Cause: The LOCAL_TRANSACTION transaction type is not supported by this database.

Response: Use NO_TRANSACTION instead.

208 Cannot disconnect from the database *DataBaseName.*

Cause: The connection between the adapter and database cannot be closed.

Response: Check the SQL exception in the Integration Server error logs and database error messages for details.

209 Cannot create writer with file path *FilePathName* or JDBC Log.

Cause: JDBC log file creation failed.

Response: Check that the log file path has the correct watt.adapter.JDBC.JDBCLogFile setting.

210 Cannot unlock webMethods OEM JDBC driver license.

Cause: The OEM version of the DataDirect Connect for JDBC driver cannot be unlocked with the key “webMethods”.

Response: Check that the driver is the OEM version and that the key is "webMethods".

306 The adapter does not support Ordered Notification for this database *DataBaseName*. Please select another service or notification template.

Cause: Ordered Notifications are not supported on this database.

Response: Use a BasicNotification or StoredProcedure Notification instead of OrderedNotification.

307 The adapter does not support Automatic Notification for this *DataBaseName*. Please select another operation template.

Cause: The Automatic Notification (InsertNotification, UpdateNotification, or DeleteNotification) is not supported for this database.

Response: Use a BasicNotification or StoredProcedure Notification instead of InsertNotification, UpdateNotification, or DeleteNotification.

308 There must be at least one expression for the SELECT statement.

Cause: You did not specify any rows using the SELECT tab for the configured service.

Response: Add rows to the SELECT tab.

309 Select at least one column from the main table.

Cause: There is no column specified from the table.

Response: Add at least one column of the main table under the SELECT tab.

310 The database vendor *VendorName* does not support the database trigger condition.

Cause: The WHEN trigger condition does not apply to this database.

Response: Do not use the WHEN tab with the notification.

311 The connection is not available for *NotificationCallbackName*.

Cause: There is no connection available in the connection pool.

Response: Check the adapter connection and contact your administrator to increase the number of connections.

312 Cannot commit the transaction to the database. *DataBaseName*.

Cause: The transaction commit failed.

Response: Check the SQL exception in the Integration Server error logs and database error messages for details.

313 Cannot read data for the output field *OutputFieldName.*

Cause: I/O error occurred when trying to read byte sequence data.

Response: Call Customer Care for assistance.

314 Cannot set data for the input field *InputFieldName*.

Cause: The input field value is not numeric.

Response: Change to a numeric input value.

316 Cannot execute the SQL statement *SQLStatement.*

Cause: SQL statements failed to execute.

Response: Check the SQL exception in the Integration Server error logs and database error messages for details.

318 Cannot get the list of catalogs.

Cause: Catalog information for the database cannot be retrieved.

Response: Check the SQL exception in the Integration Server error logs and database error messages for details.

319 Cannot get the list of table columns.

Cause: Column information for the database object cannot be retrieved.

Response: Check the SQL exception in the Integration Server error logs and database error messages for details.

320 Cannot get the list of stored procedures.

Cause: Stored procedure information for the database cannot be retrieved.

Response: Check the SQL exception in the Integration Server error logs and database error messages for details.

321 Cannot get the list of schemas.

Cause: Schema information for the database cannot be retrieved.

Response: Check the SQL exception in the Integration Server error logs and database error messages for details.

322 Cannot get the list of tables.

Cause: Table information for the database cannot be retrieved.

Response: Check the SQL exception in the Integration Server error logs and database error messages for details.

326 This database does not support stored procedure calls using JDBC stored procedure escape syntax.

Cause: Stored procedure calls are not supported by this database.

Response: Do not use stored procedure services.

327 This notification is not ready to be enabled.

Cause: Configuration of the notification is not complete.

Response: See Chapter 5, "Adapter Notifications" on page page 157 for complete instructions for configuring notifications.

331 The String for the input field *InputFieldName* does not contain a parsable number.

Cause: The input String value is not numeric.

Response: Change to a numeric input String value.

333 You must have the Record ID column listed under the SELECT tab.

Cause: You did not configure the Record ID column.

Response: Add the Record ID column using the SELECT tab for the Basic Notification.

334 A notification procedure can only have a single result set.

Cause: You configured more than one result set for the Stored Procedure Notification.

Response: Rewrite the stored procedure and configure only one result set.

335 A notification procedure can only have a single Oracle REF Cursor.

Cause: You configured more than one Oracle REF Cursor for the Stored Procedure Notification.

Response: Rewrite the stored procedure and configure only one Oracle REF Cursor.

336 If you choose Only Once Notification, you must also check the Delete Selected Records box to avoid duplicate document warning messages.

Cause: The Delete Selected Records box is not checked.

Response: Check the Delete Selected Records box.

337 The notification should not be configured on a connection with *TransactionType*.

Cause: Notification is configured with connection of transaction type other than LOCAL_TRANSACTION.

Response: Reconfigure the notification using LOCAL_TRANSACTION.

338 The data mapping for field *FieldName* is not supported.

Cause: The data mapping is not correct.

Response: For a list of supported data type mappings, see "Data Type Mapping" on page 169.

339 The number of Base Name characters used in Notification Configure tab must not exceed *MaximumCharacterLength*.

Cause: The Base Name is too long.

Response: Refer to the message itself and shorten the Base Name using the Notification Configure tab.

401 Cannot execute AS/400 command *CommandName.*

Cause: The AS/400 environment may not be correct.

Response: Check the command and error message.

402 Cannot create file on AS/400.

Cause: An error occurs when the adapter creates the file on the AS/400 system.

Response: Check the file name and AS/400 file system.

403 Cannot create trigger on AS/400.

Cause: An error occurs when the adapter creates a trigger on the AS/400 system.

Response: Check whether there is already a trigger with this name. Also check whether the user has rights to create the trigger.

404 Cannot drop trigger on AS/400.

Cause: Errors occur when the adapter drops a trigger from the AS/400 system.

Response: Check whether the trigger exists.

501 *BaseName* is not a valid name. For the notification on AS/400, the name of the source table, buffer table and trigger should not exceed 10 characters.

Cause: The names are longer than 10 characters.

Response: Change the base name so that the names of buffer table and trigger are 10 characters or less.

webMethods

APPENDIX A

Data Type Mapping

JDBC Data Type to Java Data Type Mappings

Each column in the database table is assigned a SQL type. The JDBC driver maps each SQL data type to a JDBC data type. The JDBC Adapter then maps each JDBC data type to one or more Java data types that are used as the input or output of the adapter service or notification.

The following table shows the JDBC data type to Java data type mappings. You can map each JDBC data type to a set of Java data types by choosing one from the set. The JDBC data type you select during configuration will then map to the input or output of the adapter service or notification.

See "JDBC Data Type to Java Data Type Mappings" on page 170 for a list of data types for which the Integration Server has some constraints.

Note: The JDBC Adapter does not support the datalink DB2 data type when using the adapter with DB2 for AS/400 or DB2 for OS/390.

Note: The JDBC Adapter does not support user-defined data types, Oracle PL/SQL collections, or Oracle PL/SQL records.

Note: You can use the BigDecimal data type with the JDBC Adapter. However, the webMethods Developer does not support the BigDecimal data type. This means that the Developer will not correctly display a BigDecimal data type result and you cannot enter a value of this data type. However, the JDBC Adapter will process the BigDecimal data type correctly.

JDBC Data Type	Java Data Type
BIT	java.lang.Boolean java.lang.String
TINYINT	java.lang.Byte java.lang.Integer java.lang.String
SMALLINT	java.lang.Short java.lang.Integer java.lang.String
INTEGER	java.lang.Integer java.lang.String
BIGINT	java.lang.Long java.lang.String

JDBC Data Type	Java Data Type
FLOAT	java.lang.Double java.lang.String
REAL	java.lang.Float java.lang.String
BOOLEAN	java.lang.Boolean java.lang.String
DOUBLE	java.lang.Double java.lang.String
NUMERIC	java.math.BigDecimal java.lang.String
DECIMAL	java.math.BigDecimal java.lang.String
CHAR	java.lang.String java.lang.Character
VARCHAR	java.lang.String
LONGVARCHAR	java.lang.String
DATE	java.sql.Date java.util.Date
TIME	java.sql.Time java.util.Date java.lang.String
TIMESTAMP	java.sql.Timestamp java.util.Date java.lang.String
BINARY	byte array (byte [])
VARBINARY	byte array (byte[])
LONGVARBINARY	byte array (byte[])
*CLOB	java.sql.CLOB java.lang.String
*BLOB	java.sql.BLOB byte array
OTHER	java.lang.Object java.lang.String

Note: If you plan to use CLOBs or BLOBs that exceed 4 KB, be sure to use an Oracle OCI driver or Oracle 10g Thin driver.

JDBC Data Type to Java Data Type Mapping Constraints

The Integration Server has some constraints when mapping JDBC data types to Java data types.

If you select one of the following Java data types, the data type will map exactly to the Input/Output tab in the Developer:

- java.lang.String
- java.lang.Byte
- java.lang.Boolean
- java.lang.Character
- java.lang.Double
- java.lang.Float
- java.lang.Integer
- java.lang.Long
- java.lang.Short
- java.util.Date
- java.math.BigDecimal
- java.math.BigInteger
- java.lang.Object

Those data types not included in this list will map to java.lang.Object. In these cases, if the JDBC data type you specify is for input, you will need to pass in the object with the selected Java data type. If the JDBC type is for output, you can cast the object to the selected Java data type.

SQL Data Type to JDBC Data Type Mappings

For the mappings from SQL data types to JDBC data types, see your vendor's specifications.

webMethods

APPENDIX B

Built-In Transaction Management Services

Transaction Management Overview

This appendix provides an overview and examples of using transactions. It describes how the Integration Server supports the built-in services used to manage explicit transactions for your JDBC Adapter services in the WmART package. See "Built-In Transaction Management Services" on page 183 for descriptions of each of the specific built-in transaction management services that can be used with the WmART package.

For information about other built-in services available with the JDBC Adapter, see the *webMethods Integration Server Built-In Services Reference*, which is available from the Help menu of the webMethods Developer.

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.

Transaction Types

The Integration Server supports the following kinds of transactions:

- A *local transaction* (LOCAL_TRANSACTION) which is a transaction to a resource's local transaction mechanism
- An *XAResource transaction* (XA_TRANSACTION) which is a transaction to a resource's XAResource transaction mechanism

The Integration Server can automatically manage both kinds of transactions, without requiring the adapter user to do anything. The Integration Server uses a container-managed (implicit) transaction management approach based on the J2EE CA specification, and also performs some additional connection management. This is because adapter services use connections to create transactions. For more information about implicit transactions, see "Implicit and Explicit Transactions" on page 175.

However, there are cases where adapter users need to explicitly control the transactional units of work. Examples of these cases are provided in "Implicit and Explicit Transactions" on page 175.

To support transactions, the Integration Server relies on a built-in J2EE-based Transaction Manager. The Transaction Manager is responsible for beginning and ending transactions, maintaining a transaction context, enlisting newly connected resources into existing transactions, and ensuring that local and XAResource transactions are not combined in illegal ways.

Note: The Transaction Manager *only* manages operations performed by adapter services. Other kinds of services are beyond its control.

Important! You cannot step or trace a flow that contains a transacted adapter service.

XA Transactions

If an XA transactional connection throws an exception during a service transaction and the exception results in an inconsistent state, you may need to resolve the transaction using the tools provided with the database.

For information about using the Integration Server to manage XA transactions, see the *webMethods Integration Server Administrator's Guide*.

Implicit and Explicit Transactions

Implicit transactions are automatically handled by the Integration Server 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 both local and XAResource 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 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 any number of XA_TRANSACTION connections but no more than one LOCAL_TRANSACTION connection. If your flow contains adapter

services that use more then one LOCAL_TRANSACTION connection, you must use explicit transactions, which are described in the next section.

See "Implicit and Explicit Transaction Examples" on page 177 for implicit transaction examples.

For more information about designing and using flows, see the *webMethods Developer User's Guide*.

For more information about transaction types, see "Transaction Management of JDBC 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 any number of XA_TRANSACTION connections but no more than one LOCAL_TRANSACTION connection.

Note: With explicit transactions, you must be sure to call either a commitTransaction() or rollbackTransaction() for each startTransaction(); otherwise you will have dangling transactions that 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()
```

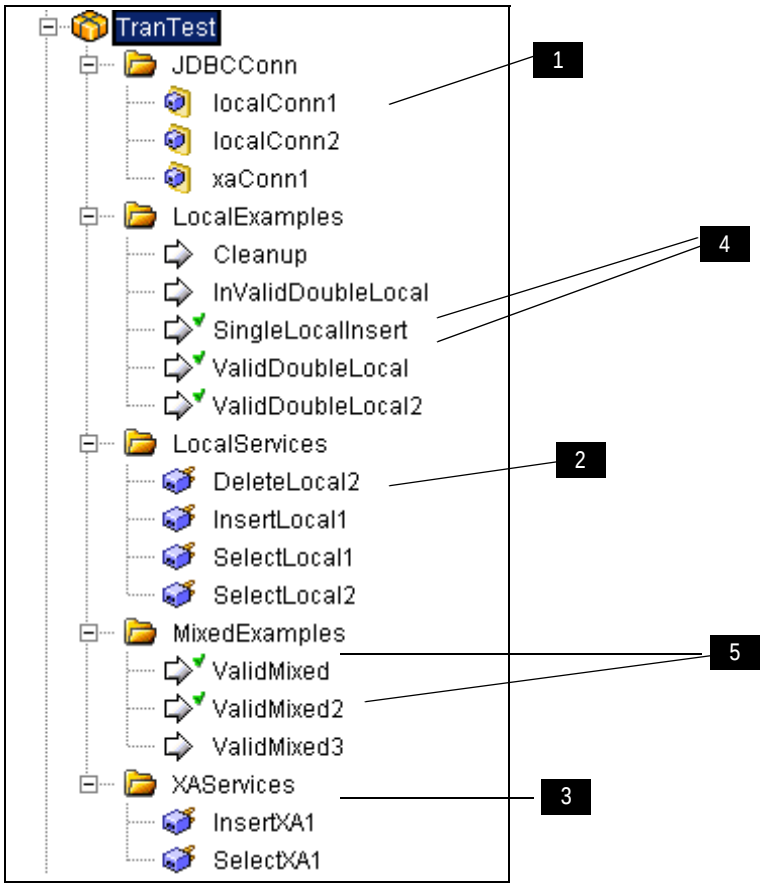
See "Implicit and Explicit Transaction Examples" on page 177 for explicit transaction examples.

For more information about designing and using flows, see the *webMethods Developer User's Guide*.

For more information about transaction types, see "Transaction Management of JDBC Adapter Connections" on page 15.

Implicit and Explicit Transaction Examples

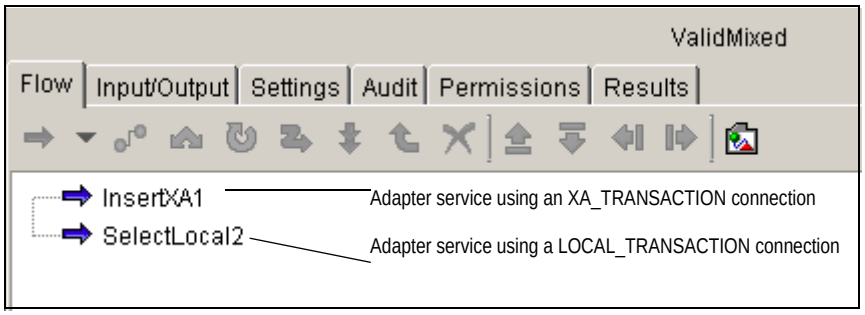
The examples in this section use the connections, services, and flows shown below and described in the table that follows.



Step	Description
1	You configured three connections: ■ localConn1: LOCAL_TRANSACTION type ■ localConn2: LOCAL_TRANSACTION type ■ xaConn1: XA_TRANSACTION
2	You configured the following adapter services which use the LOCAL_TRANSACTION connections listed in step 1 above. ■ InsertLocal1: configured to use LocalConn1 connection ■ SelectLocal1: configured to use localConn1 connection ■ SelectLocal2: configured to use localConn2 connection
3	You configured the following adapter services which use the XA_TRANSACTION connection listed in step 1 above. ■ InsertXA1: uses xaConn1 connection ■ SelectXA1: uses xaConn1 connection
4	You create the following flow examples (described in this section) using LOCAL_TRANSACTIONs: ■ SingleLocalInsert (explicit transaction). See page 180. ■ ValidDoubleLocal (explicit transaction). See page 182.
5	You create the following flow examples (described in this section) using both XA_TRANSACTIONs and LOCAL_TRANSACTIONs: ■ ValidMixed (implicit transaction). See page 179. ■ ValidMixed2 (implicit/explicit transaction). See page 181.

Flow Example: ValidMixed

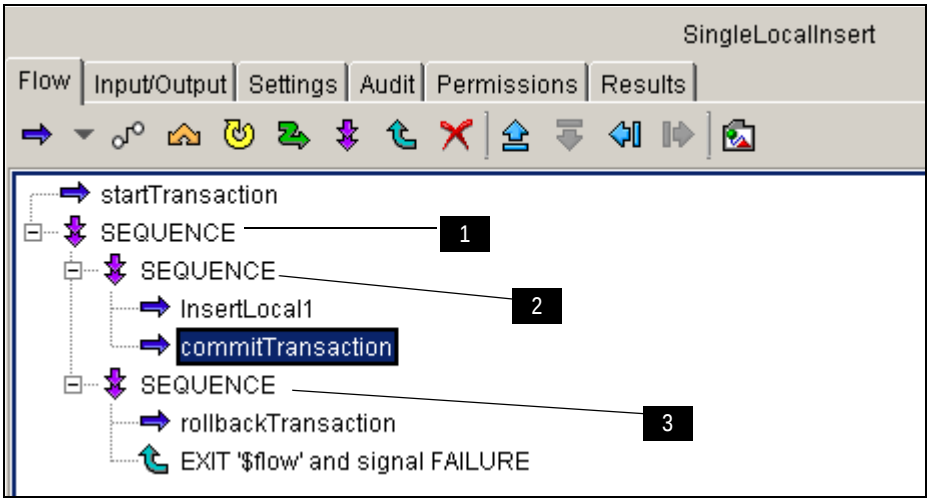
- This examples shows an Implicit Transaction.
- This flow calls:
  - One service using an XA_TRANSACTION connection (InsertXA1 service)
  - One service using a LOCAL_TRANSACTION connection (SelectLocal2 service)



Flow Example: SingleLocalInsert

- This examples shows an Explicit Transaction.
- This flow calls one adapter service (InsertLocal1) using a LOCAL_TRANSACTION connection.

This example demonstrates the correct way to set up your flow to use an explicit transaction. You use the following construct of three SEQUENCEs, which is required to insure that the explicit transaction is either committed (on success) or rolled back (on failure).



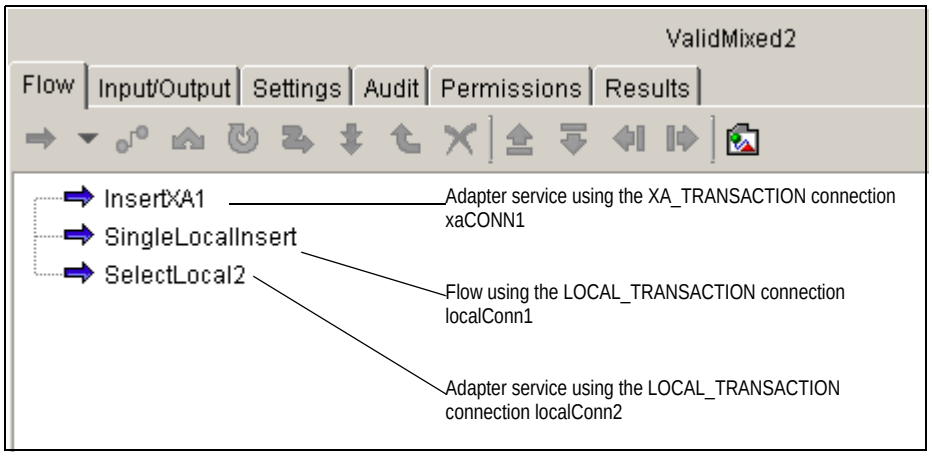
Step	Description
1	The top-level SEQUENCE will exit on success.
2	The transaction will be committed if successful, and the top-level SEQUENCE will complete.
3	This SEQUENCE is entered only if the previous SEQUENCE is unsuccessful. The transaction is rolled back and the flow exits with a status of failure.

Note that with this construct, you will not get trigger retries or a retryable exception. The EXIT statement will result in generating a Flow exception which is not retryable. To get retries, you will need to use a REPEAT step statement in your flow. For information about using the REPEAT statement, see the *webMethods Developer User's Guide*.

Flow Example: ValidMixed2

- This examples shows both an Implicit and Explicit Transaction.
- This flow calls:
  - One adapter service (InsertXA1) using an XA_TRANSACTION connection
  - One flow (SingleLocalInsert—shown in the last example on page 180) which contains its own explicit transactions and using a LOCAL_TRANSACTION connection (localConn2)
  - One adapter service (SelectLocal2) using the same LOCAL_TRANSACTION connection (localConn2) as the SingleLocalInsert flow

In this example, InsertXA1 and SelectLocal2 are registered as part of the implicit transaction. SingleLocalInsert is part of its own explicit transaction. The explicit transaction is required since you are using two different local transaction connections (localConn1 and localConn2).

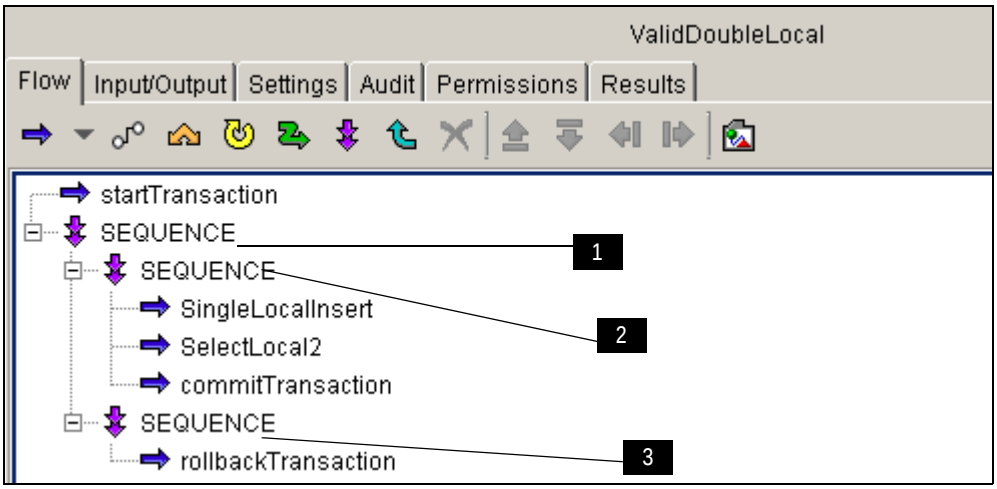


Flow Example: ValidDoubleLocal

- This example shows an Explicit transaction.
- This flow calls:
  - A flow (SingleLocalInsert) which uses the LOCAL_TRANSACTION connection localConn1
  - An adapter service (SelectLocal2) which uses the LOCAL_TRANSACTION connection localConn2

This flow shows an explicit transaction residing within another explicit transaction. The flow calls a flow and an adapter service which use different LOCAL_TRANSACTION connections. Recall that you must use an explicit transaction if you have more than one LOCAL_TRANSACTION connection.

Notice that the flow is similar to the SingleLocalInsert flow example on page 180, which uses a flow construct involving three SEQUENCEs to insure that the explicit transaction is either committed (on success) or rolled back (on failure).



Step	Description
1	The top-level SEQUENCE will exit on success.
2	The transaction will be committed if successful, and the top-level SEQUENCE will complete.
3	This SEQUENCE is entered only if the previous SEQUENCE is unsuccessful. The transaction is rolled back and the flow exits with a status of failure.

Built-In Transaction Management Services

The following sections describe each of the built-in services you can use with the wmART package.

pub.art.transaction:commitTransaction

This service commits an explicit transaction. It must be used in conjunction with the pub.art.transaction:startTransaction service. If it does not have a corresponding pub.art.transaction:startTransaction service, your flow service will receive a runtime error.

See "Transaction Management Overview" on page 174 for more information about implicit and explicit transactions.

Input Parameters

commitTransactionInput	Document. A document that contains the variable *transactionName*, described below.
transactionName	String. Used to associate a name with an explicit transaction. The *transactionName* must correspond to the *transactionName* in any pub.art.transaction:startTransaction or pub.art.transaction:rollbackTransaction services associated with the explicit transaction. This value must be mapped from the most recent pub.art.transaction:startTransaction that has not previously been committed or rolled back.

Output Parameters

None.

pub.art.transaction:rollbackTransaction

This service rolls back an explicit transaction. It must be used in conjunction with the pub.art.transaction:startTransaction service. If it does not have a corresponding pub.art.transaction:startTransaction service, your flow service will receive a runtime error.

See "Transaction Management Overview" on page 174 for more information about implicit and explicit transactions.

Input Parameters

rollbackTransactionInput	Document. A document that contains the variable *transactionName*, described below.
transactionName	String. Used to associate a name with an explicit transaction. The *transactionName* must correspond to the *transactionName* in any pub.art.transaction:startTransaction or pub.art.transaction:commitTransaction services associated with the explicit transaction. This value must be mapped from the most recent pub.art.transaction:startTransaction that has not previously been committed or rolled back.

Output Parameters

None.

pub.art.transaction:setTransactionTimeout

This service enables you to manually set a transaction timeout interval for implicit and explicit transactions. When you use this service, you are temporarily overriding the Integration Server transaction timeout interval. See "Changing the Integration Server Transaction Timeout Interval" on page 186 to change the server's default transaction timeout.

You must call this service within a flow before the start of any implicit or explicit transactions. Implicit transactions start when you call an adapter service in a flow. Explicit transactions start when you call the pub.art.transaction:startTransaction service.

If the execution of a transaction takes longer than the transaction timeout interval, all transacted operations are rolled back.

This service only overrides the transaction timeout interval for the flow service in which you call it.

Input Parameters

timeoutSeconds	Integer The number of seconds that the implicit or explicit transaction stays open before the transaction manager marks it for rollback.

Output Parameters

None.

pub.art.transaction:startTransaction

This service starts an explicit transaction. It must be used in conjunction with either a pub.art.transaction:commitTransaction service or pub.art.transaction:rollbackTransaction service. If it does not have a corresponding pub.art.transaction:commitTransaction service or pub.art.transaction:rollbackTransaction service, your flow service will receive a runtime error.

See "Transaction Management Overview" on page 174 for more information about implicit and explicit transactions.

Input Parameters

startTransactionInput	Document. A document that contains the variable *transactionName*, described below.
transactionName	String. Specifies the name of the transaction to be started. This parameter is optional. If you leave this parameter blank, the Integration Server will generate a name for you. In most implementations, it is not necessary to provide your own transaction name as input.

Output Parameters

startTransactionOutput	Document. A document that contains the variable *transactionName*, described below.
transactionName	String. The name of the transaction the service just started.

Changing the Integration Server Transaction Timeout Interval

The Integration Server 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.

webMethods

APPENDIX C

Database Driver Known Limitations

Driver Limitations

This appendix provides a high-level list of limitations and issues. For additional details, refer to your vendor documentation.

Driver	Database/Adapter IS Operating System/Platform Affected	Limitation Description
DataDirect Connect for JDBC 3.2	UDB 7.2	This driver version does not support the BLOB data types. If you try to select data from a table that has BLOB data types, you will receive the following message: `[DataDirect][DB2 JDBC Driver][DB2]AN UNSUPPORTED SQLTYPE WAS ENCOUNTERED IN POSITION 2 ON A PREPARE OR DESCRIBE OPERATION` Note that this driver does support BLOB data types using OS/390 or DB2 iSeries V5R2.
	UDB 7.2	Driver does not support XA_TRANSACTIONS using Java Transaction API (JTA). Instead, use UDB 8.1.
	UDB 7.2	Cannot insert into a BLOB column type if you use byte array as the Input Field Type. The workaround is to use the IBM drivers (DB2 app type 2 or DB2 net type 3).
	UDB 7.2 and UDB 8.1	Cannot use the CLOB data type in the OUT parameter in StoredProcedure services. Doing so throws [DataDirect][DB2 JDBC Driver][DB2]DATA TYPE/LENGTH/VALUE OF ARGUMENT 1 OF CLOBSP1 IS INVALID. The CallableStatement.getClob() does not work; instead, use the IBM driver versions (DB2 app type 2 or DB2 net type 3).
	UDB 7.2 and UDB 8.1	Cannot run a StoredProcedure service using BLOB and CLOB data types (java.sql.Blob or java.sql.Clob) as the IN parameter. Instead, use an IBM driver (DB2 app type 2 or DB2 net type 3) with UDB 8.1 to work with IN, OUT LOB parameters.

Driver	Database/Adapter IS Operating System/Platform Affected	Limitation Description
DB2 JDBC app (type 2)	AIX5.1	Cannot enable XA_TRANSACTION connections.
	Linux	Cannot create XA_TRANSACTION connections.
	UDB DB2 8.1/Sun Solaris	Cannot run a SelectSQL adapter service with table names that use special characters. Note that you can do so if you use a MicroSoft Windows NT operating system and a JDBC app (type 2) driver.
	UDB DB2 7.2	If a Stored Procedure Notification has been enabled for long periods of time, the following message is posted: `[IBM][CLI Driver][DB2/SUN] SQL1131N DARI (Stored Procedure) process has been terminated abnormally is posted. SQLSTATE=38503`
	UDB 8.1	No error message is issued when inserting a string that is larger than the size of the column defined for CHAR(N) or VARCHAR(N).
	UDB 8.1 on AIX5.1	The Integration Server crashes if the database is shut down while executing an InsertSQL adapter service using an XA_TRANSACTION connection.
DB2 Universal type 4	UDB 8.1	This driver does not support XA_TRANSACTION. Use the Universal Type 2 driver if you need XA_TRANSACTION support. White space characters are not removed from SQL Statements entered in the SQL textbox for CustomSQL or DynamicSQL services. This driver passes the SQL statements to the server exactly as entered. Be sure the SQL you enter has no extraneous white space characters, such as new lines or tabs.
	DB2 on OS/390 6 and 7	Insert, Delete, Update, Basic, and Ordered Notifications cannot be enabled if the source table contains CHAR, VARCHAR, or LONG VARCHAR columns.

Driver	Database/Adapter IS Operating System/Platform Affected	Limitation Description
DB2 JDBC net (Type3)	DB2 7 on OS/390	If you attempt to insert 20k or more records, either the system will deadlock or you will receive a timeout error.
	DB2 7.2 on OS/390	When configuring a JDBC Adapter notification in Developer (File > New > Adapter Notification), the Base Name you specify on the Notification Configure tab must be no more than 5 characters because triggers on OS/390 name cannot be more than 8 characters.
	DB2 on OS/390	Using a SelectSQL service, you cannot select a large volume of data (20k) using the CLOB data type.
	UDB 7.2	The driver does not write to the JDBC log, even when the log option is enabled. The workaround is to create an empty log file. To do this, use the Integration Server Administrator and select Settings > Extended > Edit Extended Settings and type: `watt.adapter.JDBC.JDBCLogFile= c:\log.txt`
	UDB 7.2	If you run a BatchUpdateSQL service that has no records that match your search criteria, you will receive an error; you must have at least one record that matches the criteria to execute successfully.
	UDB 8.1	No error message is issued when inserting a string that is larger than the size of the column defined for CHAR(N) or VARCHAR(N).
Informix Driver for JDBC Version 2.21 type 4	Informix 7.31 and 9.x	This driver does not support multiple results sets. If you configure the adapter to use multiple result sets, all result rows will be stored in the first Result Set you specified when you configured the adapter.
	Informix 7.31 and 9.x	With Informix 9.3 and 9.4 using XA_TRANSACTION, you cannot update LONGVARCHAR data type columns with a null value.
	Informix 7.31 and 9.x	With Informix 9.3 and 9.4 using XA_TRANSACTION, you cannot update BOOLEAN data type columns with a NOT NULL value.

Driver	Database/Adapter IS Operating System/Platform Affected	Limitation Description
jCONNECT 5.5 type 4	Sybase 11.x and 12.x	A Sybase column using a BIT data type does not allow NULL values due to driver behavior. This means that if you insert a NULL or ? (variable) value when you run an InsertSQL service, the driver converts this column value to false and inserts the NULL value for the column into the database. Stored procedures support only one result set.
Oracle JDBC Drivers (Oracle JDBC Thin and Oracle JDBC/OCI)	All supported Oracle databases	The NUMBER and NUMBER(n,m) Oracle data types map to java.math.BigDecimal in all JDBC Adapter services by default.
	Oracle 8.0.5	When mapping a date data type to java.util.Date using the InsertSQL adapter service, you receive an "ORA-1024 Invalid data type in OCI call" exception. As a workaround, map the date to java.sql.Timestamp.
	HP-UX 11i	Be sure to apply the HP-UX 11i Quality Pack (June 2002) and the PHSS_26138 on HP-UX 11i before configuring the JDBC Adapter connection using an OCI driver; otherwise, it throws an “Unresolved symbol :gethrtime (code)” error.
	Oracle 8.0.5	When connecting to an Oracle 8.0.5 server using the OCI driver and trying to Insert BLOB and CLOB data types, you receive a "ORA-01461: can bind a LONG value only for insert into a LONG column." error.
	HP-UX	If all adapter notifications are enabled for more than 18 hours, you receive the error: `OCI-21503: program terminated by fatal error OCI-04030: out of process memory when trying to allocate 20056 bytes.`
	All supported databases	BLOB and CLOB data types cannot be used in a table definition when configuring JDBC Adapter notifications.
Oracle JDBC OCI Driver	All supported databases	You must set an environment variable before you can configure connections. See “Setting the Environment Variable for Oracle JDBC OCI Drivers” on page 67 for details.
Oracle JDBC Thin Driver	All supported databases	BLOB/CLOB data type objects are limited to 4 KB. For larger BLOB/CLOB objects, use the Oracle JDBC/OCI driver or Oracle 10g Thin driver.

Driver	Database/Adapter IS Operating System/Platform Affected	Limitation Description
SQL Server 2000 Driver for JDBC Version 2.2.0019	Microsoft SQL Server 2000	■ When running the SelectSQL adapter service using the "not null real" type, the "Value cannot be converted to requested type" error is thrown. This is a driver issue for both the DataDirect Connect for JDBC and the Microsoft SQL Server 2000 Driver for JDBC. ■ This driver does not support retrieving table names from a database when the database's name contains special characters. ■ This driver must have Sun's JDK 1.3 package javax.sql.* in the Integration Server CLASSPATH before you can enable the adapter connection. If this package is missing, the adapter throws the error message "unable to configure connection manger javax/sql/DataSource." One solution is to copy the adapter's jdbcxsql.zip file, which is located in the *IntegrationServer_directory*\packages\WmJDBCAdapter\code\jars directory, to the *IntegrationServer_directory*\lib\jars directory, placing the required files in a location where the Integration Server can include them in its CLASSPATH.
SQL Server 2005 Driver for JDBC Version 1.0.809.102	Microsoft SQL Server 2005	■ This driver returns incorrect data type TEXT, IMAGE, and NTEXT for MS SQL data types VARCHAR(max), VARBINARY(max) and NVARCHAR(max) respectively. ■ This driver returns invalid JDBC data type for MS SQL UNIQUEIDENTIFIER data type.
Teradata V2R5 (Type 4)	All supported databases.	This driver does not support the following services and notifications: ■ BatchInsertSQL or BatchUpdateSQL services ■ InsertNotifications, UpdateNotifications, DeleteNotifications, and OrderedNotifications
	All supported databases	If you use the @ character in a table or column name, you will receive the following syntax error: `Expected something between the word 'SP$CHAR#TABLE' and '@'." while using in Insert service on Teradata.`

webMethods

APPENDIX D

Determining Whether to Use the WmDB Package or the JDBC Adapter

Overview

webMethods provides two methods for accessing databases:

- WmDB package to use for prototyping, design, or temporary database access.
- webMethods JDBC Adapter to use for all enterprise, mission critical applications.

This appendix describes use cases to help you determine when to use the WmDB package and when to use the webMethods JDBC Adapter.

When to Use the WmDB Package

The following lists use cases where it would be better to use the WmDB package rather than the JDBC Adapter. Use the WmDB package when:

- You need immediate results from the execution of a SQL query. The WmDB package allows you to run quick SQL queries directly from the Integration Server Administrator or DSP pages.
- You have a need for ad hoc introspection of database objects, such as tables, views, aliases, synonyms, or stored procedures.
- You want to customize management of the database connection pool.
- You have a limited number of tasks that need to access a database.
- You are required to integrate with a proprietary or a non-mainstream database, and you want to perform a quick, easy compatibility test before exploring an implementation that uses the JDBC Adapter. The JDBC Adapter supports all JDBC-compliant database implementations and is certified on specific mainstream databases.
- You are developing a throw-away test scenario.

When to Use the webMethods JDBC Adapter

You will want to use the JDBC Adapter in most cases. The following lists use cases that describe the types of situations when you should use the JDBC Adapter rather than the WmDB package:

- Your system is likely to be expanded and modernized. This is typical for enterprise, mission critical applications.
- You require better database performance. The performance of the JDBC Adapter is superior to that of the WmDB package, particularly for batch operations. For example, to insert data into a database using the WmDB package you use the pub.db:insert service; you supply the items to insert using a document list as input, and the

pub.db:insert service processes each item in the document list sequentially without using JDBC standard facilities.

- You require or plan to use notification features.
- You need automatic and efficient database connection pool management.
- You require XA transactionality.
- You need to use JDBC standard data types, including BLOB and CLOB.
- You need to be able to configure services rather than hard code them.
- You need to use role-based security. The JDBC Adapter allows the separation of the database administrator and development environments as needed. The WmDB package requires Administrator privileges.
- You want a configurable and structured user interface. The JDBC Adapter uses template-based configuration that is more structured and easier to use. The JDBC Adapter uses metadata that helps protect the user's investment even if technology changes.
- You want to manage your database connection using webMethods Manager.

Index

Symbols

A

B

C

D

T

U

V

W

X